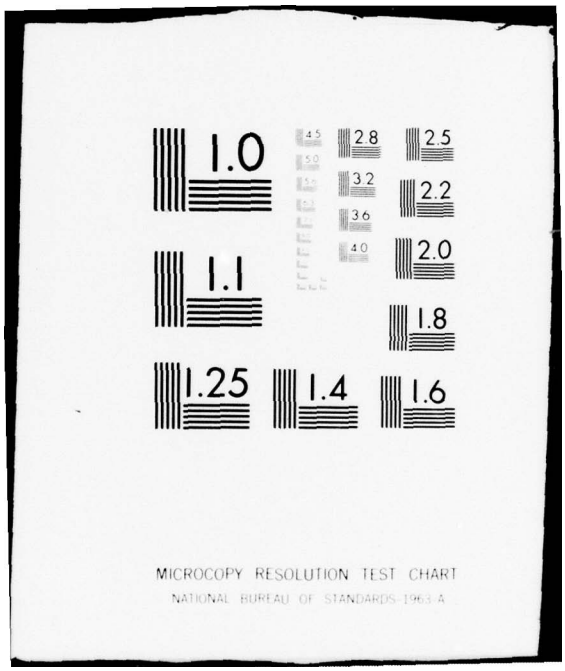
AD-A057 962 STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE F/G 12/1
NUMERICAL ALGORITHMS FOR NONLINEARLY CONSTRAINED OPTIMIZATION.(U)
APR 78 M T HEATH DAHC04-76-G-0185
UNCLASSIFIED STAN-CS-78-656 NL

1 OF 2
AD
A057962


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

NUMERICAL ALGORITHMS FOR NONLINEARLY CONSTRAINED OPTIMIZATION

by

Michael Thomas Heath

STAN-CS-78-656
APRIL 1978

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

(6) NUMERICAL ALGORITHMS FOR NONLINEARLY CONSTRAINED OPTIMIZATION.

(9) Doctoral thesis,
by

(10) Michael Thomas Heath

(11) Apr 77

(12) 150 p.

(14) STAN-CS-78-656

(15) DAHC04-76-G-0185,
EY-76-S-03-0326

Research supported in part under U. S. Army Research grant DAHC04-76-G-0185 and by Department of Energy contract EY-76-S-03-0326 PA #30.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER STAN-CS-78-656	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) NUMERICAL ALGORITHM FOR NONLINEARLY CONSTRAINED OPTIMIZATION		5. TYPE OF REPORT & PERIOD COVERED Technical (thesis) April 1978 6. PERFORMING ORG. REPORT NUMBER STAN-CS-78-656
7. AUTHOR(s) Michael T. Heath		8. CONTRACT OR GRANT NUMBER(s) DAHC04-76-G-0185
9. PERFORMING ORGANIZATION NAME AND ADDRESS Stanford University Computer Science Department Stanford, Ca. 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709		12. REPORT DATE April 1978 13. NUMBER OF PAGES 148
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Releasable without limitations on dissemination.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This dissertation is concerned with the development and numerical implementation of algorithms for solving finite dimensional optimization problems. Special emphasis is given to robustness, by which is meant the ability of an algorithm to cope with adverse circumstances, whether due to pathologies of a particular problem or to the shortcomings of finite precision computer arithmetic. A uniform framework is developed in which a common set of techniques may be applied to all of the standard problems of optimization. The algorithms are based on Newton-like methods implemented in a robust manner by means of hybrid,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

BLOCK 20 continued:
curved line searches and stable linear algebra techniques. Developed first in the context of systems of nonlinear equations, non-linear least squares, and unconstrained minimization, the algorithms are combined and extended to include problems with equality or inequality constraints. Constrained problems are handled by means of separate line searches in the range and null spaces of the matrix of constraint normals. The classical Lagrangian is modified to allow the same Newton-like methods to be applied to inequality constraints. Test results are presented which show the validity and promise of the methods developed in this dissertation.

ABSTRACT

This dissertation is concerned with the development and numerical implementation of algorithms for solving finite dimensional optimization problems. Special emphasis is given to robustness, by which is meant the ability of an algorithm to cope with adverse circumstances, whether due to pathologies of a particular problem or to the shortcomings of finite precision computer arithmetic. A uniform framework is developed in which a common set of techniques may be applied to all of the standard problems of optimization. The algorithms are based on Newton-like methods implemented in a robust manner by means of hybrid, curved line searches and stable linear algebra techniques. Developed first in the context of systems of nonlinear equations, nonlinear least squares, and unconstrained minimization, the algorithms are combined and extended to include problems with equality or inequality constraints. Constrained problems are handled by means of separate line searches in the range and null spaces of the matrix of constraint normals. The classical Lagrangian is modified to allow the same Newton-like methods to be applied to inequality constraints. Test results are presented which show the validity and promise of the methods developed in this dissertation.

ACKNOWLEDGMENTS

The author owes a debt of gratitude to a number of individuals and institutions for support of various kinds during his graduate education and in the research for and writing of this dissertation. I thank my thesis director, Professor Gene Golub, not only for his advice and encouragement during my thesis research, but also for originally encouraging me to attend Stanford. I have had an unusually active reading committee which included Professor Golub, Dr. John Greenstadt, Professor Joseph Oliger, and Dr. Margaret Wright, and I thank each of them for their careful reading of the manuscript. Their detailed comments and criticisms led to many significant improvements and were greatly appreciated. Among the many teachers from whom I have benefitted, I single out three who directly influenced my choice of doctoral research area: Professor Golub and Professor Alston Householder, from whom I learned numerical linear algebra, and Professor Richard Tapia, from whom I learned nonlinear analysis and optimization. The generous financial support of the Fannie and John Hertz Foundation during the bulk of my career at Stanford is gratefully acknowledged, as is the use of the excellent computer facilities at the Stanford Linear Accelerator Center. I would also like to thank Jerri Rudnick for her expert and expeditious typing. Finally, I thank my wife Mona for her patience and understanding while this work was in progress.

TABLE OF CONTENTS

Table of Contents (cont.)

CHAPTER 1. INTRODUCTION AND BACKGROUND

1.1 Introduction

This thesis is concerned with algorithms for solving finite dimensional optimization problems. Our primary concern is with practical questions of algorithm development and numerical implementation. Special emphasis is given to robustness, by which is meant the ability of an algorithm to cope with adverse circumstances, whether due to pathologies of a particular problem or to the shortcomings of finite precision computer arithmetic. The possession of nice theoretical properties is a necessary but far from sufficient condition for an algorithm to be effective for a wide class of practical problems. This fact should be well known to anyone who has tried a textbook algorithm, programmed in a straightforward manner, on real-world problems.

The typical assumptions upon which theoretical analyses are usually based--nonsingularity or positive definiteness of matrices, constraint qualifications, etc.--are frequently violated by real problems, either in principle or in their discretized, digital computer counterparts. Such contingencies must be explicitly taken into account if they are to be handled intelligently. Many of the algorithms employed in this thesis reduce, in favorable circumstances, to an existing algorithm whose theoretical analysis is well known. Therefore, a key tool in our approach is the question "What can go wrong with this algorithm?" In this way we try to deal not only with the question of what to do until the local convergence theorem applies, but what to do even if it never applies. We are also concerned with the efficiency of algorithms but not at the expense of numerical stability, since getting poor results rapidly is of dubious value.

All this does not mean that we advocate an ad hoc bag of tricks to handle many different special cases. Rather, we develop a uniform framework in which a common set of techniques may be applied to all the standard problems of optimization. This viewpoint is very much in the spirit of recent trends in numerical software in that it facilitates modular programming and allows users to solve many different kinds of problems with a single subroutine. The algorithm at which we ultimately arrive will solve problems in unconstrained optimization, systems of nonlinear equations, nonlinear least squares, and general nonlinear programming (nonlinear equality and inequality constraints) with little more overhead than similar algorithms designed especially for any one of these.

The remainder of this chapter is concerned with necessary background material such as notation and definitions, classes of problems to be considered, optimality conditions, tools from computational linear algebra, and a discussion of goals in algorithm design. In Chapter 2 algorithms are developed for unconstrained optimization problems and for nonlinear equations and least squares. In Chapter 3 problems having equality or inequality constraints are considered. Chapter 4 contains test results for a computer program, based on the algorithms developed in Chapters 2 and 3, applied to a collection of test problems representing each of the problem classes. There is also an appendix in which operation counts are given comparing different linear algebra techniques for implementing the algorithms.

1.2 Notation

This work is concerned with numerical analysis and optimization, two areas whose notational conventions are not always compatible. We employ

a compromise, but perhaps one biased a bit toward the former. For the most part we use the notation given on pp. ix-x of Gill and Murray (1974a), which is summarized in Table 1. All scalars, vectors, and matrices are real. All vectors are column vectors unless transposition is explicitly indicated, and the inner product of two vectors is written as $x^T y$. Usually, but not always, upper case letters are used for matrices, lower case letters for vectors and scalars. The identity matrix is denoted by I and its i^{th} column by e_i. Inequalities between vectors are to be interpreted as holding componentwise. The Euclidean norm, $\|x\| = (x^T x)^{\frac{1}{2}}$, will be the only norm used. Ordinary approximate equality between numerical quantities will be denoted by $\approx$. The symbol $\overset{x}{=}$ is reserved to indicate linear least-squares problems. Thus $Ax \overset{x}{=} b$, where A is an $n \times m$ matrix and x and b are vectors, represents the problem $\min_x \|Ax-b\|^2$. In statements of algorithms the symbol $:=$ is used in the sense of replacement, as in some programming languages. Machine precision is the smallest floating point number ϵ on a given computer such that $1 + \epsilon > 1$.

If a sequence $\{x^k\} \subseteq \mathbb{R}^n$ converges to x^* we say that the convergence rate is first order (or linear) if there is a constant scalar β, $0 < \beta < 1$, and an integer $k_0 \geq 0$ such that

$$\|x^{k+1}-x^*\| \leq \beta\|x^k-x^*\| \quad , \quad \text{for all} \quad k \geq k_0 \quad .$$

The convergence rate is said to be second order (or quadratic) if there is a constant scalar β and an integer $k_0 \geq 0$ such that

$$\|x^{k+1}-x^*\| \leq \beta\|x^k-x^*\|^2 \quad , \quad \text{for all} \quad k \geq k_0 \quad .$$

TABLE 1

Symbol	Meaning
n	number of independent variables; dimension of domain.
x	n-vector of independent variables.
f	objective function to be minimized; $f: \mathbb{R}^n \to \mathbb{R}$.
g	gradient vector of f; $g(x) = \nabla f(x)$; $g_i(x) = \partial f(x)/\partial x_i$.
G	$n \times n$ Hessian matrix of f; $G(x) = \{\partial^2 f(x)/\partial x_i \partial x_j\}$.
m	number of constraints (equations or inequalities).
c	m-vector of constraints; $c: \mathbb{R}^n \to \mathbb{R}^m$.
A	$n \times m$ matrix whose j^{th} column is $\nabla c_j(x)$; $A(x) = \{\partial c_j(x)/\partial x_i\}$ = transposed Jacobian of c.
G_i	$n \times n$ Hessian matrix of $c_i(x)$.
λ	m-vector of Lagrange multipliers.
L	Lagrangian function; $L: \mathbb{R}^{n+m} \to \mathbb{R}$; $L(x,\lambda) = f(x) + \lambda^T c(x)$.
w	gradient (with respect to x) of Lagrangian; $w(x,\lambda) = g(x) + A(x)\lambda$.
W	Hessian (with respect to x) of Lagrangian; $W(x,\lambda) = G(x) + \sum_{i=1}^{m} \lambda_i G_i(x)$.

Finally, the convergence rate is said to be superlinear if

$$\frac{\|x^{k+1}-x^*\|}{\|x^k-x^*\|} \to 0 \quad \text{as} \quad k \to \infty \ .$$

In stating arithmetic operation counts for algorithms, one unit is defined as a multiplication or division paired with an addition or subtraction, and only the dominant term will be given. For example, Gaussian elimination for solving a system of linear equations of order n requires about $n^3/3$ arithmetic operations, or, if the coefficient is unimportant in a given context, we say it is an $O(n^3)$ algorithm.

Because analytic formulas for derivatives are often inconvenient, impractical or costly to obtain (and in any case would have to be evaluated in finite precision arithmetic), throughout this thesis we assume that all indicated derivatives may be only approximate and not necessarily very accurate. Occasional exceptions occur in certain theoretical results which require exact derivatives, but these are explicitly noted in the text. Some specific approximation techniques are discussed in Section 2.6.

1.3 Classes of Problems Considered

We will be concerned with algorithms for solving the five broad classes of problems listed below. All functions involved are assumed to be as smooth as necessary to support the theory upon which the algorithms are based. This usually means the existence of one or two continuous derivatives depending on context. Note that we do not assume any special properties such as linearity, separability, or sparseness.

1. Nonlinear equations (NLEQ):

   Given $c: \mathbb{R}^n \to \mathbb{R}^n$ find $x^* \in \mathbb{R}^n$ such that $c(x^*) = 0$.

2. Nonlinear least squares (NLLS):

   Given $c: \mathbb{R}^n \to \mathbb{R}^m$, $m \geq n$, find $x^* \in \mathbb{R}^n$ such that $\varphi(x^*) = \min\{\varphi(x): x \in \mathbb{R}^n\}$, where $\varphi(x) = \frac{1}{2}\|c(x)\|^2 = \frac{1}{2}\, c(x)^T c(x)$.

3. Unconstrained optimization (UCON):

   Given $f: \mathbb{R}^n \to \mathbb{R}$ find $x^* \in \mathbb{R}^n$ such that $f(x^*) = \min\{f(x): x \in \mathbb{R}^n\}$.

4. Equality constrained optimization (ECON):

   Given $f: \mathbb{R}^n \to \mathbb{R}$ and $c: \mathbb{R}^n \to \mathbb{R}^m$, $m \leq n$, find $x^* \in \mathbb{R}^n$ such that $f(x^*) = \min\{f(x): c(x) = 0\}$.

5. Inequality constrained optimization (ICON):

   Given $f: \mathbb{R}^n \to \mathbb{R}$ and $c: \mathbb{R}^n \to \mathbb{R}^m$ find $x^* \in \mathbb{R}^n$ such that $f(x^*) = \min\{f(x): c(x) \leq 0\}$.

The indicated abbreviations will be used in referring to these problems throughout this thesis.

Each of the first three problems can be thought of as a special case of the other two in the following senses:

1. Problems NLLS and UCON can be converted into NLEQ by seeking a stationary point (zero of the gradient) of φ or f.
2. Problem NLEQ can be taken as the special case of NLLS having $m = n$.
3. Problem NLLS is UCON with a particular form for the function to be minimized.

As these relationships suggest, computational techniques for the three problems are similar in spirit, and development of algorithms for

them has proceeded more or less in parallel. However, the three problems are not strictly equivalent and their similarities should not be pushed too far. In particular the following differences are important:

1. Techniques for solving NLEQ will not necessarily be successful for NLLS or UCON since the stationary point found could be a saddlepoint or a maximum rather than a minimum.
2. For problem NLLS one must generally be content with finding a local minimum of $\varphi(x)$, but such a point does not solve NLEQ unless $\varphi(x^*) = 0$. Moreover, $\nabla\varphi(x) = A(x)c(x) = 0$ does not necessarily imply $c(x) = 0$.
3. The special form of the objective function φ in NLLS causes its gradient and Hessian to have a special structure, not enjoyed by those of a general function f, which can be exploited in algorithms designed specifically for NLLS.

With regard to item 3 above it should be added that the greater generality of UCON over NLLS is perhaps of less practical importance than it might appear, since many functions to be minimized which occur in practice (and certainly the overwhelming majority of unconstrained test problems to be found in the literature) are in fact sums of squares. This is probably due on the one hand to the fact that one of the most common optimization problems is the matching of theory with experiment--that is, the fitting of observed data to a model by minimizing a smooth norm of the discrepancy. On the other hand more general objective functions are likely to require constraints as a natural part of the problem since functions of arbitrary form are often unbounded below or lose physical meaning in some regions. In any case we will treat the three problems as closely related but distinct.

Note that we do not have a separate category for the most general nonlinear programming problem having both equality and inequality constraints. This is because algorithms for handling the two kinds of constraints are clearer if developed separately and it will be obvious how to combine the two for a problem having both equalities and inequalities.

Several concepts relevant to problems 3-5 will be needed later. A point x satisfying the constraints, if any, is called a feasible point. The feasible set is the set of all feasible points. If there are no constraints the feasible set is all of R^n. A feasible point x^* is a local minimum if $f(x^*) \leq f(x)$ for all feasible x in some neighborhood of x^*. A feasible point x^* is a global minimum if $f(x^*) \leq f(x)$ for all feasible x. Note that problems 3-5 are stated for global minima. In practice it is extremely difficult to find or verify global minima, so we will content ourselves with algorithms for identifying local minima.

At a feasible point x for problem ICON a constraint c_i is said to be active if $c_i(x) = 0$. For problem ECON all constraints are considered active at a feasible point. A feasible point x for either ECON or ICON is called regular if $\hat{A}(x)$ has full rank, where the "hat" notation indicates that only active constraints are included. At a regular feasible point x the tangent space to the active constraint manifold is given by $M_x = \{s:\hat{A}^T(x)s = 0\}$. For problem UCON we take M_x to be R^n for all x.

1.4 Optimality Conditions

The above problems are not all stated in a form immediately suited to computational solution. What is needed is a characterization of the solution in computationally verifiable terms which are amenable to established

algorithms, such as Newton's method, designed for equation solving. This is the motivation for the following optimality conditions.

Theorem 1.1. Sufficient (necessary) conditions that a regular feasible point x^* be a local minimum for one of the following problems are

UCON: $g(x^*) = 0$ and $G(x^*)$ is positive (semi) definite.

ECON: There is a $\lambda^* \in \mathbb{R}^m$ such that $w(x^*,\lambda^*) = 0$ and $W(x^*,\lambda^*)$ is positive (semi) definite on M_{x^*}.

ICON: There is a $\lambda^* \in \mathbb{R}^m$ such that $w(x^*,\lambda^*) = 0$, $\lambda^* \geq 0$, $\lambda^{*T} c(x^*) = 0$, and $W(x^*,\lambda^*)$ is positive (semi) definite on M_{x^*}.

(Recall from the summary of notation in Table 1 that $w(x,\lambda)$ and $W(x,\lambda)$ are the gradient and Hessian of the Lagrangian function $L(x,\lambda) = f(x) + \lambda^T c(x)$.)

Proof. See Luenberger (1973), pp. 224-226 and 233-235.

Note that for each problem the conditions involve a stationary point of some functional (hence, an equation to be solved) and a criterion for determining if that stationary point is in fact a local minimum. This provides both a basis for algorithms seeking to solve these problems and a means to confirm computationally that a solution has been found. One of our goals in designing algorithms will be to make natural use of these conditions, for instance in termination criteria.

1.5 Linear Algebra

Many advances in optimization algorithms have gone hand-in-hand with improved techniques in computational linear algebra. The use of orthogonal factorizations in linear and nonlinear least squares, various factorizations

of symmetric matrices in unconstrained optimization and quadratic programming, rank-one matrix updating techniques in quasi-Newton methods, and projection matrices in nonlinear programming are but a few examples. The algorithms discussed in this thesis make vital use of a number of modern computational techniques for the linear algebra problems which arise. We briefly review some of these in this section.

1.5.1 Orthogonal Factorizations. Let A be an $n \times m$ matrix with $n \geq m$. Then there is an orthogonal matrix Q (i.e., $Q^TQ = I$) such that

$$QA = \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{1.1}$$

where R is $m \times m$ and upper triangular (see, for example, Stewart (1973), p. 214). We write Q in partitioned form as

$$Q^T = (Y,Z), \tag{1.1a}$$

where Y is $n \times m$ and Z is $n \times (n-m)$. Suppose, for the moment, that A has rank m. Then

a. R is nonsingular.

b. The columns of Y and Z form orthonormal bases for the range and null spaces, respectively, of A.

c. The matrices YY^T and ZZ^T are orthogonal projectors onto the range and null spaces, respectively, of A.

d. The unique solution to the linear least-squares problem $Ax \cong b$ (i.e., $\min_x \|Ax-b\|^2$) is given by the solution x^* to the triangular linear system $Rx = Y^Tb$ with the residual vector

given by $b-Ax^* = ZZ^Tb$ and the approximating vector by $Ax^* = YY^Tb$.

Making use of this last fact, the factorization (1.1) offers a much more stable method for computing least-squares solutions than the traditional normal equations

$$A^TAx = A^Tb$$

and, as we shall see, allows greater flexibility in diagnosing and coping with the rank deficient case. A simple example showing the inadequacy of the normal-equation method is given by the matrix

$$A = \begin{pmatrix} 1 & 1 \\ \epsilon & 0 \\ 0 & \epsilon \end{pmatrix},$$

where ϵ is of the order of magnitude of machine precision. Then

$$A^TA = \begin{pmatrix} 1+\epsilon^2 & 1 \\ 1 & 1+\epsilon^2 \end{pmatrix},$$

so that A^TA is exactly singular in floating point arithmetic. It should be noted, however, that the price for the greater stability of orthogonalization methods is that more computational work is required. For the $n \times m$ matrix A, if $n \approx m$ then the two approaches have similar operation counts, but if $n \gg m$ then orthogonalization requires about twice as many arithmetic operations as the normal equations. Excellent discussions of solving linear least-squares problems by orthogonalization and by other methods are given by Stewart (1973), Lawson and Hanson (1974),

and Van Loan (1976).

There are a number of algorithms for computing orthogonal factorizations. The most convenient approach for our purposes is described by Householder (1958) and was popularized by Golub (1965). An Algol program is given in Businger and Golub (1965). In this algorithm the matrix A is reduced to upper-triangular form by premultiplying it by a sequence of elementary Hermitian matrices which annihilate the subdiagonal elements of successive columns. The elementary Hermitian matrices, often called Householder transformations or reflections, have the form

$$U = I - \frac{1}{\beta} uu^T \quad ,$$

where $\beta = \frac{1}{2} u^T u$. It can be verified readily that such a matrix satisfies $U = U^T = U^{-1}$, so that it is orthogonal and symmetric. If a is any vector (e.g., a column of A) and we take

$$\alpha = \text{sign } (a_k) \cdot (\sum_{j=k}^{n} a_j^2)^{\frac{1}{2}} \quad ,$$

$$u = (0,\ldots,0,a_k + \alpha,a_{k+1},\ldots,a_n)^T \quad ,$$

and

$$\beta = \alpha u_k \quad ,$$

then

$$Ua = (I - \frac{1}{\beta} uu^T)a = (a_1,\ldots,a_{k-1},-\alpha,0,\ldots,0)^T \quad .$$

Since the effect of U on any vector is determined solely by the vector u, the matrix U need not be explicitly formed. If b is any vector, then $Ub = b - \frac{1}{\beta} u(u^T b)$ so that only the inner product $u^T b$ and a vector subtraction (not a full matrix multiplication) are required to compute

the transformed vector. Furthermore, the first $k-1$ components of b are unchanged by the transformation; in particular, any existing zero components in these positions are preserved. Thus, proceeding successively from the first to last columns of A, a sequence of Householder transformations is generated such that

$$U_m \cdot U_{m-1} \cdots U_2 U_1 A = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

with $Q = U_m \cdots U_1$ orthogonal (we may take $U_m = I$ if $m = n$). If the linear least-squares problem $Ax \cong b$ is being solved by this method, then ordinarily the vector b (as well as A) would be transformed successively by the individual U_i in the efficient manner indicated earlier, so that the product matrix Q need not be formed explicitly. However, if the bases for the range and null spaces of A given by the columns of the matrices Y and Z are needed explicitly, then the product $Q = U_1 \cdots U_m$ may be accumulated very efficiently by premultiplying the identity matrix successively by the U_i, working backward from U_m through U_1. This need for full orthogonal bases for both range and null spaces is one reason for our use of Householder transformations rather than Gram-Schmidt orthogonalization. The latter procedure produces a basis for the range space of A only, and furthermore, the columns of Y which it generates may deviate significantly from orthogonality (see Wilkinson (1965), p. 243).

So far we have assumed that A has full rank. If this is not the case, say A has rank $k < m$, then the orthogonal-triangular factorization (1.1) exists and the algorithm given above for computing it works. However, the triangular matrix R is no longer nonsingular, so that an attempted

least-squares solution would break down in the back-substitution phase (this corresponds to a singular matrix A^TA in the normal equations).

In the rank deficient case, it is not clear what is meant by a solution since there are infinitely many vectors x which minimize the least-squares residual $\|Ax-b\|^2$. Out of this set of solutions we follow Rosen (1964) in singling out the following two as particularly useful. A vector x^* is the <u>minimum solution</u> to $Ax \cong b$ if $\|Ax^*-b\| \leq \|Ax-b\|$ for all x, and $\|x^*\| \leq \|x\|$ for all x such that $\|Ax^*-b\| = \|Ax-b\|$. A vector x^* is a <u>basic solution</u> to $Ax \cong b$ if $\|Ax^*-b\| \leq \|Ax-b\|$ for all x, and x^* has at most k nonzero components, where k is the rank of A. The minimum solution is unique while a basic solution, in general, is not. Minimum and basic solutions may be described as least squares solutions of smallest norm and fewest parameters, respectively. We now consider algorithms for computing them.

A basic solution to $Ax \cong b$ results from choosing any k linearly independent columns of A, solving the corresponding full rank $n \times k$ problem, and setting the remaining $m-k$ components to zero. One particular selection of linearly independent columns can be accomplished during the orthogonal factorization process by interchanging columns of A so that at each step the unreduced column of largest norm is brought into the next position to be reduced. In theory this column pivoting procedure causes k linearly independent columns to be selected for reduction first and gives a factorization of the form

$$QAP = \begin{pmatrix} R & S \\ 0 & 0 \end{pmatrix} , \qquad (1.2)$$

where P is the permutation matrix which performs the column interchanges

and R is $k \times k$, upper triangular and has diagonal elements $|r_{11}| \geq |r_{22}| \geq \cdots \geq |r_{kk}| > 0$ (see, for example, Lawson and Hanson (1974), p. 12). In practice the situation is not so clear cut because in finite precision arithmetic the diagonal elements r_{ii} and the unreduced lower right-hand block become small but, due to rounding errors, remain nonzero. Since the rank k of A is generally unknown in advance, it must be estimated numerically; this is usually done by truncating the factorization when some criterion for smallness of the remaining unreduced columns or for the condition of R is met. Heuristic rules which are reasonable, if not entirely rigorous, are usually adequate for this purpose. (See Karasalo (1974) for an example of work along this line.) However, it should be borne in mind that this issue is greatly complicated by the fact that the rank of a given matrix may not be well determined numerically in that a small perturbation to the matrix may cause the rank to change (see, for example, Golub, Klema and Stewart (1976)).

Once the factorization (1.2) has been obtained then a basic solution to $Ax \cong b$ is given by $x = P\begin{pmatrix} y \\ z \end{pmatrix}$, where the k-vector y is the solution to the triangular system $Ry = Y^T b$ and the (m-k)-vector z has all its components zero. This is the basic solution we use henceforth.

We now turn to the computation of the minimum solution to $Ax \cong b$. An efficient way of computing this is by means of the complete orthogonal factorization of A (see Peters and Wilkinson (1970) or Lawson and Hanson (1974), p. 13). Continuing from the factorization (1.2), the block S can be annihilated by applying a sequence of Householder transformations on the right, resulting in the complete orthogonal factorization

$$QAV = \begin{pmatrix} \bar{R} & 0 \\ 0 & 0 \end{pmatrix}, \qquad (1.3)$$

where both Q and V are orthogonal and $\bar{R}$ is again $k \times k$, upper triangular and nonsingular. (Note that the permutation matrix P is included, along with the product of the Householder transformations, in the matrix V.) The minimum solution to $Ax \cong b$ is now given by $x = V\begin{pmatrix} y \\ z \end{pmatrix}$, where the k-vector y is the solution to the triangular system $\bar{R}y = Y^T b$ and the $(m-k)$-vector z has all its components zero.

Thus, although the minimum solution has a certain theoretical attractiveness, it is somewhat more costly to compute than the basic solution, and the latter serves well for most practical purposes. (See Golub and Pereyra (1976), pp. 308-309, for a quantitative comparison.) Of course, the two solutions coincide in the full-rank case. For future reference we note here that the basic and minimum solutions are those given by the generalized inverses

$$A^B = P\begin{pmatrix} R^{-1} & 0 \\ 0 & 0 \end{pmatrix}Q$$

and

$$A^+ = V\begin{pmatrix} \bar{R}^{-1} & 0 \\ 0 & 0 \end{pmatrix}Q \quad ,$$

respectively. The latter is known as the Moore-Penrose pseudoinverse (see Penrose (1955)). As with nonsingular square systems of equations, explicit computation of these inverses is neither necessary nor desirable for solving the linear system.

Another factorization of A which leads to the minimum solution is the singular value decomposition

$$A = U\begin{pmatrix} \Sigma \\ 0 \end{pmatrix}V^T \quad , \tag{1.4}$$

where $\Sigma = \text{diag}(\sigma_i)$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > \sigma_{k+1} = \cdots = \sigma_m = 0$, and

U and V are orthogonal matrices of order n and m, respectively (see Forsythe and Moler (1967), p. 5). The diagonal entries σ_i of Σ, called the singular values of A, are the nonnegative square roots of the eigenvalues of A^TA, and the columns of U and V are orthonormal eigenvectors of AA^T and A^TA, respectively. An efficient and stable algorithm for computing the singular value decomposition (without explicit formation of A^TA or AA^T) is given by Golub and Reinsch (1970). In terms of this factorization the minimum solution to $Ax \cong b$ is given by

$$x = \sum_{i=1}^{k} \frac{u_i^T b}{\sigma_i} v_i \quad ,$$

where u_i and v_i denote the columns of U and V, respectively. Although the singular value decomposition requires roughly twice as much work as the orthogonal-triangular factorization, it provides a great deal more information about the matrix A. In particular, the relative magnitudes of the diagonal entries of Σ give a more clear-cut indication of the rank of A than is available from a triangular factorization. The singular value decomposition also provides an alternate expression for the Moore-Penrose pseudoinverse,

$$A^+ = V(\Sigma^+,0)U^T \quad ,$$

where $\Sigma^+ = \text{diag}(\sigma_1^{-1},\ldots,\sigma_k^{-1},0,\ldots,0)$.

1.5.2 Symmetric Factorizations. Let B be an $n \times n$ real symmetric matrix. Suppose, for the moment, that B is positive definite. Then it has a factorization of the form

$$B = LDL^T \quad , \tag{1.5}$$

where D is a diagonal matrix having positive diagonal elements and L is a unit lower triangular matrix. This factorization can be computed by a simple variant of the familiar Cholesky algorithm. (See Forsythe and Moler (1967) or Stewart (1973).) This algorithm has among its attractive features that no pivoting is required for stability; the elements of L are bounded even without row and column interchanges (see Forsythe and Moler (1967), p. 115). This not only makes the algorithm simple and efficient, but also allows the factorization to be updated easily when a symmetric rank-one change is made to the matrix B (see Gill, Golub, Murray and Saunders (1974)). The factorization (1.5) can be used to solve linear systems of the form $Bx = c$ by successively solving the triangular systems $LDy = c$ and $L^Tx = y$.

All these nice properties are critically dependent on the positive definiteness of B. If B is not positive definite, then the factorization (1.5) does not exist (at least not with D a real diagonal matrix having positive diagonal entries), and the Cholesky algorithm breaks down. If the diagonal elements of D are allowed to have arbitrary sign (including zero), then a factorization like (1.5) exists for a wider (but still not all-inclusive) class of matrices, but even in these cases may be quite unstable to compute unless row and column interchanges are performed.

A much more satisfactory generalization of (1.5), first suggested by W. Kahan in 1965, is to allow D to be block diagonal with diagonal blocks of order one or two. An algorithm for computing such a factorization was given by Bunch and Parlett (1971) and has since been improved in efficiency by Bunch and Kaufman (1977a), which should be consulted for the

details of the following discussion. An Algol program is given in Bunch, Kaufman and Parlett (1976). The basic idea of the algorithm, called the block-diagonal-pivoting method, is to perform symmetric Gaussian elimination but allow 2×2 block pivots as well as the usual 1×1 scalar pivoting operations. At a typical step the remaining unreduced matrix $B^{(i)}$ has the form

$$B^{(i)} = \begin{pmatrix} X & C^T \\ C & Y \end{pmatrix},$$

where X is nonsingular and may be of order $k = 1$ or 2. The next reduced matrix is then given by

$$B^{(i-k)} = Y - CX^{-1}C^T.$$

(Note that the indices run backward.) The choice of order k for X depends on predicted growth in the elements of $B^{(i-k)}$. For simplicity we have omitted the permutations of rows and columns necessary to assure nonsingularity of X and avoid error growth in the reduced matrices, but in practice the pivot search employed is of critical importance to the efficiency and stability of the algorithm. Since simple partial pivoting on a single column is known to be unstable in this context, Bunch and Parlett (1971) originally used a complete pivoting strategy at a cost of $O(n^3)$ comparisons. Bunch and Kaufman (1977a) have since developed a partial pivoting strategy which examines at most two columns at each step in the reduction so that it requires only $O(n^2)$ comparisons and is at the same time almost as stable as complete pivoting. The arithmetic operation count for the algorithm is of order $n^3/6$, which is the same as the Cholesky algorithm for positive definite matrices and half that of ordinary Gaussian

elimination for general matrices.

Barwell and George (1976) give empirical comparisons of the block-diagonal-pivoting method with several other methods for solving symmetric linear systems. The Bunch-Kaufman program proved to be at least as good as any other algorithm for indefinite matrices. Moreover, it is only very slightly inferior to the Cholesky algorithm when the latter is applicable.

In summary, for a symmetric matrix B there is a numerically stable algorithm for efficiently computing the factorization

$$P^T B P = L D L^T \quad , \tag{1.6}$$

where P is a permutation matrix, L is a unit-lower-triangular matrix, and D is a block-diagonal matrix having diagonal blocks of order one or two. If B happens to be positive definite, then $P = I$ and D is diagonal. This factorization is used as before to solve symmetric linear systems. In addition, it provides other useful information about B, in particular its inertia. The inertia of a real symmetric matrix is a triple of integers consisting of the numbers of positive, negative, and zero eigenvalues of the matrix. According to Sylvester's Law of Inertia (see Gantmacher (1959), Vol. 1, p. 297), the inertia of a real symmetric matrix is invariant under nonsingular congruence transformations. Thus, the block-diagonal matrix D in (1.6) has the same inertia as the matrix B. The inertia of D is easily determined by inspection. The block-diagonal-pivoting procedure is such that the determinant of a 2×2 block is always negative, and thus has one positive and one negative eigenvalue. Therefore, the number of positive eigenvalues of D is the number of 2×2 blocks plus the number of positive 1×1 blocks.

Similarly, the number of 2×2 blocks plus the number of negative 1×1 blocks equals the number of negative eigenvalues. In this way it can be determined (at least up to rounding error in computing D) whether B is in fact positive definite without actually computing its eigenvalues. The factorization (1.6) has other uses as well which will be encountered later.

One disadvantage of the block-diagonal factorization (1.6) is that the permutations required for stability, and possible changes in the block structure of D, make it considerably more complicated to update the factorization after a symmetric rank-one change in the matrix B. This problem has been addressed by Bunch and Kaufman (1977b) for adding or deleting rows and columns and by Sorensen (1977) for changes of the form $B + uu^T$. There are other symmetric factorizations which might be more amenable to such updating, such as the tridiagonal factorization of Parlett and Reid (1970) and Aasen (1971), but these do not provide the wealth of information about B which (1.6) makes available, information we shall put to good use in optimization algorithms.

1.6 Design Goals

As a summary of our motivating philosophy we list the following design goals as desirable of any algorithm for optimization problems.

1. Rate of local convergence. To be competitive an algorithm should have at least a superlinear asymptotic convergence rate.
2. Robustness with respect to starting point. An iterative algorithm should be able to converge to the desired solution without

requiring a close estimate for the solution and should be able to circumvent singularities and similar troubles along the way.

3. Stability of linear algebra. In solving linear algebra subproblems, only those matrix factorizations which are known to exist and can be stably computed should be used. For example, Cholesky factorization of a symmetric matrix should not be relied on if there is a chance the matrix is not positive definite. Generally, normal equations should not be used for least-squares solutions, especially if there is a chance of rank deficiency.

4. Size of linear systems. The systems of linear equations to be solved should be as small as possible in order to minimize computational overhead and accumulated rounding errors. For example, some algorithms for minimizing a function of n variables subject to m equality constraints employ linear systems of order n or even $n + m$, when the natural dimension for this problem is $n - m$, the actual number of degrees of freedom.

5. Amount of computational overhead. The linear algebra techniques should be as efficient as possible so long as numerical stability is maintained. For example, factorizations should be updated, if possible, rather than recomputed when the matrix changes. Eigenvalue and singular value decompositions should not be used when a less costly factorization serves as well.

6. Natural use of optimality conditions. This is important not only in guiding the algorithm in seeking a solution but also in computationally verifying that a point to which it has converged is in fact a solution to the problem.

7. Discrimination between different kinds of stationary points. A minimization algorithm should be able to steer clear of or descend from saddlepoints and maxima.

8. Insensitivity to choice of parameters. The performance of an algorithm should not be critically dependent on a judicious choice of parameters such as weighting coefficients, penalty constants, tolerances, etc., and "fine tuning" should not be required for individual problems.

9. Application to special problems. When applied to a class of problems having special properties, a general purpose algorithm should reduce to a stable and reasonably efficient method for that particular kind of problem. For example, a general nonlinear programming code should be able to solve systems of nonlinear equations or inequalities, unconstrained optimization problems, least-squares problems, complementarity problems, etc., without too much more overhead than special algorithms for these problems would require.

10. Derivatives required. Although higher derivatives of problem functions may reasonably be assumed to exist for theoretical analysis, these can be tedious or highly impractical to derive, program, and debug. Therefore, practical algorithms should require that only first derivatives (at most) be analytically defined by the user and second derivatives, if needed, should be approximated.

Of course, it is unlikely that any single algorithm could fully attain all of these objectives; indeed, some of them conflict with others.

Nevertheless, they serve as a useful guide in choosing among various alternatives in algorithm design and convey the spirit of our approach to the subject.

CHAPTER 2. UNCONSTRAINED PROBLEMS

2.1 Newton's Method for Problem NLEQ

Newton's method, in its many variants and guises, has come to mean more a philosophy for treating nonlinear problems than a specific algorithm. It serves as paradigm for many, perhaps most, rapidly convergent iterative algorithms. The popularity of Newton's method is doubtless due to its excellent asymptotic convergence rate, which in normal circumstances is second order. Despite its pervasive influence, however, Newton's method is seldom used in its purest form. One reason is that for general problems only local convergence can be guaranteed, so that unless the Newton iteration is begun near the solution, convergence cannot be expected. In practice this flaw turns out not to be a mere lack of theoretical certainty, but actually manifests itself often as a divergent sequence of iterates or other catastrophic failure. Thus, in absence of a good initial estimate for the desired solution, a naive implementation of Newton's method can be unreliable and care must be taken to avoid disaster while outside the natural domain of local convergence of the basic algorithm. This motivation has led to many variants of Newton's method designed to improve its robustness, and often to reduce its computational overhead as well. In this chapter we consider Newton's method and a number of these variants in the context of specific classes of problems to which they may be applied. We develop a unified view of the various proposals leading to a strategy which is consistent for all problem classes and which can be efficiently implemented. A thorough discussion of Newton's method and a number of its variants is contained in Ortega and Rheinholdt (1970).

Both the virtues and shortcomings of Newton's method are to be found in its motivating philosphy, that of local linearization. Linearization is a case in point of a familiar theme in numerical analysis, that of replacing a difficult problem by a sequence of easier ones whose solutions approach that of the original problem. We first consider problem NLEQ in which we seek a solution x^* to the system of nonlinear equations $c(x) = 0$, where $c: \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable. Given an initial guess x for the solution, define the affine function $T_x: \mathbb{R}^n \to \mathbb{R}^n$ by $T_x(s) = c(x) + A^T(x)s$, where $A^T(x)$ is the Jacobian of c at x (see Table 1). Since $T_x(s)$ is merely a truncated Taylor series for $c(x+s)$ about the point x, $T_x(s)$ approximates $c(x+s)$ well near x. Thus, the solution to the problem $T_x(s) = 0$ may be taken as a new, and presumably better, approximate solution to the original problem since it tends to make $c(x+s) \approx 0$. The process is now repeated for the new x, and so on until the original equation is satisfied as accurately as desired. We summarize this procedure as

Algorithm 2.1 (for NLEQ)

Initial data : x

Repeat until convergence tolerance met

1. Evaluate $c(x)$, $A(x)$
2. Solve $A^T s = -c$
3. $x := x + s$

The local convergence properties of this algorithm can be summarized as follows.

Theorem 2.1. If $c(x^*) = 0$ and $A(x^*)$ is nonsingular, then there is an open neighborhood N of x^* such that if the initial x lies in N, then Algorithm 2.1 (using exact derivatives) converges to x^* with order 2.

Proof. See Ortega and Rheinboldt (1970), p. 312.

The main difficulty with this theorem, as with all local convergence results, is that the neighborhood N may be (and in practice often is) quite small, so that supplying a starting guess known to lie within N is very difficult. Another potential problem to note is the assumed nonsingularity of $A(x^*)$. Even when this assumption is valid (as in practice it usually, though not always, is) the Jacobian may well be singular at a given point outside N, causing Step 2 of the algorithm to fail. On the other hand, singularity of $A(x^*)$ does not necessarily cause nonconvergence, but usually does degrade the rate of convergence to first order. In any case the important point is that if singularity of $A(x)$ causes theoretical difficulties, then, as with all calculations using finite precision computer arithmetic, near singularity or ill-conditioning of $A(x)$ can be expected to cause numerical difficulties in carrying out the steps of Algorithm 2.1.

All the above difficulties can be understood in terms of the local linearization upon which the algorithm is based. The trouble is that the extent of the region in which $T_x(s)$ is an adequate approximation to $c(x+s)$, sometimes called the region of trust, is unknown and may be rather small. In particular the solution to $T_x(s) = 0$ may lie outside the region of trust. In this case the computed step s may have little relation to the solution of the original problem, quite possibly making $x+s$ a worse approximation to x^* than x is. Clearly this situation could lead to

divergence or at best erratic progress toward a solution. The practical implication to be drawn is that a computed step s which is "large" should be regarded with suspicion. Obviously the concept of an acceptable magnitude for the step length can have only a relative, not an absolute meaning. Moreover, it is highly problem and data dependent in that if c is very nonlinear at x then the region of trust for the linear approximation is correspondingly small, whereas, at the other extreme, Newton's method solves a linear problem in a single iteration from an arbitrary starting point. Nevertheless, these considerations have led to a number of alternative algorithms in which Newton steps of excessive magnitude are avoided if deemed unreliable. Before exploring these issues further, let us consider Newton's method in some other contexts.

2.2 Newton's Method for Problems UCON and NLLS

For problem UCON, the unconstrained minimization of a scalar valued function f, a local quadratic approximation to f at a point x is given by the truncated Taylor series

$$f(x+s) \approx f(x) + g(x)^T s + \frac{1}{2} s^T G(x) s \quad .$$

The stationary point of this quadratic form in s is seen by differentiation to be given by the solution to the linear system $G(x)s = -g(x)$. Thus we arrive at Newton's method for problem UCON:

Algorithm 2.2 (for UCON)

Initial data : x

Repeat until convergence tolerance met

1. Evaluate $g(x)$, $G(x)$

2. Solve $Gs = -g$

3. $x := x + s$

Note that this is just Algorithm 2.1 applied to the problem $g(x) = 0$, and therefore Algorithm 2.2 has similar properties, both good and bad. It does have the advantage that the matrix G is symmetric, which should help in solving the linear system in Step 2. By the optimality condition for UCON, if x is near a local minimum of f, then $G(x)$ can be expected to be positive definite so that the Cholesky factorization is applicable. However, far from the solution $G(x)$ may be indefinite (either singular or having negative eigenvalues), which implies warnings similar to those concerning the possible singularity or near singularity of $A(x)$ in Algorithm 2.1. Moreover, Algorithm 2.2 as stated may converge to a stationary point of f which is not a local minimum. These difficulties will be addressed later.

For problem NLLS,

$$\nabla\varphi(x) = A(x)c(x)$$

and

$$\nabla^2\varphi(x) = A(x)A^T(x) + \sum_{i=1}^{m} c_i(x)G_i(x) \quad .$$

In applying Algorithm 2.2 to the minimization of φ the linear system to be solved in Step 2 whould thus be

$$(AA^T + \sum_{i=1}^{m} c_iG_i)s = -Ac \quad .$$

Note that $\nabla^2\varphi(x)$ is a sum of two terms, the first of which involves first derivatives only. The second term, however, is a sum of m separate Hessians of scalar functions. This plethora of second-order partial derivatives could be a nuisance to derive and expensive to evaluate, so a common device, going back to Gauss, is to drop the second term and use the approximation $\nabla^2\varphi(x) \approx A(x)A^T(x)$ in the algorithm. The justification is that problem NLLS often arises in the context of curve fitting; if the fit is reasonably good (as it should be if the model is to represent the data meaningfully, then the neglected second-order term is small (at least near the solution) relative to the first-order term which is retained. In any case this simplification leads to the linear system

$$AA^Ts = -Ac \quad .$$

Note, however, that this is just the system of normal equations for the linear least-squares problem

$$A^Ts \cong -c \quad ,$$

which, as discussed in Section 1.5.1, is best solved directly via orthogonalization. These considerations are summarized in the Gauss-Newton algorithm for NLLS:

Algorithm 2.3 (for NLLS)

Initial data : x

Repeat until convergence tolerance met

1. Evaluate $c(x)$, $A(x)$
2. Solve $A^Ts \cong -c$ in the least squares sense
3. $x := x + s$

Note that this algorithm looks exactly like Algorithm 2.1 except that the linear system in Step 2 may now be overdetermined. If $m = n$ and A is nonsingular, then the two algorithms are identical, which is to be expected since in this case $\varphi(x^*) = 0$ and the Gauss approximation to the Hessian is exact at x^*. However, in the general overdetermined case $(m > n)$ we will usually not have $\varphi(x^*) = 0$, and the Gauss approximation may not be adequate. In particular if $\varphi(x^*) \neq 0$, then Algorithm 2.3 may not be locally convergent and, if it does converge, the convergence rate is only first order. Nevertheless, the Gauss-Newton algorithm--not the true Newton algorithm--will be the basis for our algorithms for NLLS because there is little point in discussing NLLS separately from the more general problem UCON if the full Hessian is computed. Algorithms designed for UCON can always be applied to NLLS in the large residual $(\varphi(x^*) >> 0)$ case. Thus, the Newton step for problem NLLS refers to the vector s given by a least-squares solution to $A^T s \cong -c$. (However, there <u>are</u> algorithms which approximate the full Hessian while still exploiting the structure of problem NLLS. See Dennis, Gay and Welsch (1977).)

2.3 Damped Newton Method

As we saw in Section 2.1, taking a full Newton step when far from a solution is potentially dangerous. Large Newton steps may be avoided in two ways: restricting the length of the Newton step while retaining its direction, or abandoning the Newton step in favor of some other direction.

The primary rule in the first approach is that no step will be accepted unless it results in improvement over the current estimate for the solution. If the full Newton step violates this rule then a shorter

step in the same direction, $x + \alpha s$, where $0 < \alpha < 1$, is tried. Improvement is measured by means of a scalar valued function, called a merit function, which has a local minimum at the solution and is convex in a neighborhood of the solution. Thus, if ψ is the merit function, it is insisted that the descent condition

$$\psi(x+\alpha s) < \psi(x)$$

is satisfied at each step. For problem UCON, f is a natural choice for the merit function, while φ is an appropriate choice for problems NLEQ and NLLS.

It should be noted that descent is a local strategy and not necessarily the best strategy globally. In special circumstances enforcing descent could lead to a less direct route to a solution or even cause divergence, although it is difficult to see how such situations could be anticipated in practice. On the whole, enforcing descent would appear much more likely to help than hurt.

For reducing the functional ψ, starting at a given point x, a direction s is called a descent direction if $\nabla\psi(x)^T s < 0$. The significance of this concept is that if s is a descent direction, then the descent condition is satisfiable by taking α small enough.

Lemma 2.1. For problems UCON, NLEQ and NLLS the Newton direction is a descent direction (under conditions noted).

Proof. UCON: $\nabla f(x)^T s = -g^T G^{-1} g < 0$,

provided G is positive definite.

NLEQ: $\nabla\varphi(x)^T s = -c^T A^T A^{-T} c < 0$,

provided A is nonsingular.

NLLS: $\nabla\varphi(x)^T s = -c^T A^T (AA^T)^{-1} Ac < 0,$

provided A has full rank.

Remark. Note that this result does not require that G and A^T be the actual Jacobians of g and c. They could, in fact, be any matrices satisfying the stated properties and used in the linear system for computing the Newton step.

Thus, as long as the matrix involved has the relevant property, the Newton direction should be downhill, so that there is a neighborhood of x in which movement along s produces a lower function value. This is the justification for the damped Newton method embodied in the following algorithm (stated for UCON but applicable with obvious changes to NLEQ and NLLS).

Algorithm 2.4 (for UCON)

Initial data : x, β (a scalar, $0 < \beta < 1$)

Repeat until convergence tolerance met

1. Evaluate $g(x)$, $G(x)$
2. Solve $Gs = -g$
3. $\alpha := 1$
4. Repeat until $f(x+\alpha s) < f(x)$
    1. $\alpha := \beta\alpha$
5. $x := x + \alpha s$

The choice $\beta = 1/2$ is typical. While this algorithm avoids the obvious potential for divergence of the pure Newton method, and Lemma 2.1 might seem to give some hope for global convergence, there are still a number of difficulties:

1. The linear system in Step 2 may still be impossible to solve.

2. Despite the lemma, the computed s may not give an actual decrease in $f(x+\alpha s)$ for any α. Such failure could happen for a number of reasons. First, the hypotheses of the lemma may not be satisfied. Second, in the presence of rounding error the computed s may differ significantly from the true s of the lemma. Third, the lemma insures that s forms an acute angle with $-g$, but in practice they could be nearly orthogonal so that whether or not f decreases along s becomes problematic given the limitations of finite precision arithmetic. Finally, the interval along s in which f decreases may be extremely small, perhaps so small that the computed function value shows no change throughout it.

3. No attempt is made to diagnose a poor step in advance so that many function evaluations may be spent trying steps that are hopelessly large.

4. Excessively small steps can cause slow progress or even false convergence in the sense that the accepted α's for successive iterations become arbitrarily small and the algorithm stalls at a point which may not be a stationary point.

5. Finally, there is the philosophical objection that the use of only a small fraction of the true Newton step graphically demonstrates that the quadratic model was inadequate at the outset, thereby invalidating the motivating principle of the algorithm. In fact, the best direction along which to take tiny local steps is not the Newton direction but (by definition) the direction of

steepest descent, which in the Euclidean norm is simply $-g$ or $-Ac$.

2.4 Alternative Strategies

Several techniques have been proposed for overcoming the difficulties noted in the last section. We review these techniques in this section and in the next section show how they can be combined into various hybrid algorithms.

2.4.1 Solving the Linear System. There are two main approaches to the problem of supplying something which can be meaningfully called a solution to the linear system for the Newton step when the matrix $A(x)$ is rank deficient or $G(x)$ is not positive definite

1. Generalized matrix inversion. Ben-Israel (1965) and Fletcher (1968) have proposed the use of the Moore-Penrose pseudoinverse in Newton's method in case of rank deficiency of $A(x)$ or $G(x)$. Rather than explicity compute the pseudoinverse, one would take s to be the minimum least-squares solution to $A^T s \cong -c$, which can be computed via the complete orthogonal factorization or the singular value decomposition (see Section 1.5.1). In view of the expense of this computation it may be preferable to use some other generalized inverse for which the solution is easier to compute. An obvious candidate is the basic solution discussed in Section 1.5.1. Whatever the choice, one must still face the possibly delicate problem of numerically determining the rank of the matrix.

2. Perturbation of the Hessian to force positive definiteness.

Although the generalized inverse approach is applicable to problem UCON if one's sole intent is to solve the linear system somehow, singularity is not the only problem in this case: if $G(x)$ has any negative eigenvalues then the computed solution may not be a descent direction. Thus, it is desirable to eliminate the effect of negative as well as near-zero eigenvalues. For this reason a number of techniques have been proposed for replacing the Hessian, when it is indefinite, by a positive definite matrix. Usually an attempt is made to perturb the matrix as little as possible consistent with this objective. The new positive definite "Hessian", which we denote by $\bar{G}$, is then used in the linear system, and by Lemma 2.1 the solution is therefore a descent direction. This approach is also applicable to the symmetric matrix $A(x)A(x)^T$ of problems NLEQ and NLLS, although explicit formation of this matrix should be avoided if possible. Several suitable positive definite replacements for the Hessian have been proposed.

a. Greenstadt (1967) suggests doing an eigenvalue-eigenvector decomposition of the symmetric matrix G,

$$G = U\Gamma U^T ,$$

with $\Gamma = \mathrm{diag}(\gamma_i)$ and $U^TU = I$, then taking

$$\bar{\gamma}_i = \max\{|\gamma_i|, \epsilon, \epsilon \cdot \max_{1 \le j \le n} |\gamma_j|\} , \qquad (2.1)$$

where ϵ is the machine precision, and reassembling to get

$$\bar{G} = U\bar{\Gamma}U^T \quad .$$

Notice that $\bar{G}$ not only is positive definite but also has condition number bounded by $1/\epsilon$. Of course, the reassembly of $\bar{G}$ is not explicitly carried out since its eigenvalue-eigenvector decomposition is already available for solving the linear system. In some sense this choice for $\bar{G}$ is ideal since it gives complete information about the current Hessian, which is useful for other purposes to be discussed later, and makes full use of that information in determining the search direction. However, the required eigenvalue-eigenvector *decomposition* is costly relative to simply solving a linear system, although either cost may be minor compared to that of function and derivative evaluations.

b. Sorensen (1977) has suggested another approach which is similar in spirit but less costly to effect. First, the symmetric indefinite factorization (see Section 1.5.2)

$$G = LDL^T$$

is computed, then the eigenvalue-eigenvector decomposition

$$D = U\Gamma U^T \ ,$$

which is easy because D is block diagonal with diagonal blocks of order one or two. The $\bar{\gamma}_i$ are then defined as in (2.1) except that now the eigenvalues are

those of D rather than G. Finally, reassembly gives

$$\bar{G} = LU\bar{U}^T L^T .$$

Again $\bar{G}$ is positive definite, has bounded condition number, and is already in factored form ready for solving the linear system by back-substitution. This approach is significantly cheaper, yet still tells much about the matrix G.

c. Gill and Murray (1972) have suggested another approach along these lines in which the Cholesky algorithm is used for factorizing G, but the diagonal elements are altered during the process, if necessary, to maintain a satisfactory degree of positive definiteness. Upon completion the factorization has the form

$$\bar{G} = LDL^T = G + E ,$$

where D and E are diagonal. If G is in fact sufficiently positive definite, then no alteration is necessary, $E = 0$, and the result is the usual Cholesky factorization. Again the factorization of $\bar{G}$ is used directly in solving the linear system.

d. Levenberg (1944) and Marquardt (1963) in the context of NLLS, and Goldfeld, Quandt and Trotter (1966) in the context of UCON, suggested shifting the entire spectrum of the Hessian by a positive constant. Thus,

$\bar{G} = G + \mu I$, where the positive scalar μ is large enough so that $\bar{G}$ is positive definite. The difficulty here is in choosing an appropriate value for μ. Heuristic rules have been proposed, for example, by Fletcher (1971), Hebden (1973), and Moré (1977), but this question remains open. Levenberg and Marquardt actually used μ as a search parameter in a way to be described below, thus basing their choices on descent considerations. It is important to note that the Levenberg-Marquardt technique applied to problem NLLS does not require explicit formation of the matrix $A(x)A(x)^T$ since the system

$$(AA^T+\mu I)s = -Ac$$

is simply the set of normal equation for the linear least-squares problem (see Golub (1965), p. 213)

$$\begin{pmatrix} & A^T \\ \mu^{\frac{1}{2}} & I \end{pmatrix} s \cong \begin{pmatrix} -c \\ 0 \end{pmatrix} , \qquad (2.2)$$

which can be solved by the orthogonalization techniques discussed in Section 1.5.1.

Whichever of the above techniques is employed, the main point is that a Newton-like step can always be computed, even in circumstances where the original algorithm breaks down in theory due to rank deficiency or indefiniteness. Sufficiently near a well-behaved solution these

techniques should not be necessary, so that the algorithm reverts to its normal local convergence characteristics, including the asymptotic convergence rate. A Newton step which is not a descent direction is of little value; therefore, the Newton step for problem UCON at a point where $G(x)$ is indefinite will be taken to mean the step obtained using a perturbed positive definite matrix $\bar{G}$. Similarly, for problems NLEQ and NLLS the term "Newton step" will be used to refer to the minimum or basic solution to $A^T s \cong -c$.

2.4.2 Alternative Directions. Whenever the computed Newton step appears to be unreliable or is tried but fails to give descent, then it is necessary to have some alternative direction or directions in which to search for a function decrease if the algorithm is to continue. Two options have been proposed.

1. Negative gradient direction. The classical method of steepest descent has a poor reputation due to its (sometimes excruciatingly slow) linear asymptotic convergence rate (see Akaike (1959)). However, far from a solution--where asymptotic convergence rate is not an issue--it may be a quite reasonable alternative to Newton's method for reasons indicated at the end of Section 2.3. Steepest descent is safe in that the only limitations on its ability to attain descent are the accuracies with which the function and its gradient are evaluated. Moreover, it has the computational advantage that no Hessian or approximate Hessian is needed and no linear system must be solved. One drawback, as opposed to Newton's method, is that the negative gradient determines the

direction of steepest descent in the Euclidean norm but does not determine a nominal steplength along it. Of course, one could always simply use arbitrarily chosen trial points, but it would be nice to have a rough idea of the magnitude of an appropriate trial step to take along the negative gradient vector in advance of any line searching, since this may affect the choice of strategy in conducting the line search. A common tactic is to use a single quadratic extrapolation based on the function and gradient values at the current point or, equivalently, one step of Newton's method for the one-dimensional problem

$$\min_{\alpha} f(x-\alpha g)$$

or

$$\min_{\alpha} \varphi(x-\alpha Ac)$$

with starting point $\alpha = 0$. This gives the step

$$\alpha = \frac{g^T g}{g^T G g} \quad \text{for} \quad \text{UCON}$$

or (2.3)

$$\alpha = \frac{\|Ac\|^2}{\|A^T Ac\|^2} \quad \text{for NLEQ and NLLS ,}$$

provided the denominator in each case is positive. Note that this choice makes use of the Hessian, which is assumed to be available for other purposes anyway. Henceforth, the steepest descent step or negative gradient step is taken to mean the step obtained by scaling the negative gradient vector in this

way so that, after scaling, $\alpha = 1$ is an appropriate first step in a line search along the negative gradient. This scaling makes the Newton and gradient steps roughly comparable in magnitude, and will be helpful in combining or choosing between the two in hybrid algorithms discussed below. When the scale factor given in (2.3) is negative or undefined (or too expensive to compute), then the negative gradient may be used without scaling or it may be scaled by some other rule, such as having its length equal to unity, the radius of the trust region (see below), or the length of the Newton step (if available).

2. Negative curvature directions. For problem UCON, if the current Hessian has a negative eigenvalue γ, with corresponding eigenvector q, then $q^T G q < 0$ and the sign of q can be chosen so that $g^T q \leq 0$. Any vector q having these two properties is called a direction of negative curvature, and the above remark shows that at least one such vector exists whenever G has a negative eigenvalue. Geometrically, this means that along this direction the current point is on a hill rather than in a valley. Therefore, a function decrease should result from moving along q, as can be seen from the Taylor expansion

$$f(x+\alpha q) \approx f(x) + \alpha g(x)^T q + \frac{1}{2} \alpha^2 q^T G(x) q \quad ,$$

since the two rightmost terms are nonpositive and negative, respectively. Thus, a direction of negative curvature should be a fruitful direction in which to search for a function decrease.

Fiacco and McCormick (1968), pp. 166-167, proposed this strategy in the indefinite case but could only suggest a costly eigenvalue-eigenvector decomposition or (in this case) unstable or nonexistent Cholesky factorization as a means of computing a suitable direction of negative curvature q, so it appeared to be a theoretically interesting but impractical idea. By using the symmetric indefinite block-diagonal factorization, however, computation of q is both stable and efficient. Suppose $G(x)$ has k negative eigenvalues, $k > 0$. Then if $G = LDL^T$ is the factorization of the form (1.6), D also has k negative eigenvalues (see Section 1.5.2). Let these be $\gamma_1,\ldots,\gamma_k$ with corresponding unit eigenvectors $v_1,\ldots,v_k$. Let N be any nonempty subset of $\{1,\ldots,k\}$ and $z = \sum_{j\in N} v_j$. Then if y is the solution to the linear system $L^T y = z$, it follows that

$$y^T G y = y^T L D L^T y = z^T D z = \sum_{j\in N} \gamma_j < 0 \quad .$$

Thus, by taking $q = \pm y$ according to the sign of $g^T y$, q is a direction of negative curvature. Two obvious choices for the subset N are to use all k eigenvectors (Fletcher and Freeman (1975)) or only the eigenvector corresponding to the most negative eigenvalue of D (Sorensen (1977)). Here again the vector q represents only a direction and has no inherent natural scale. Its length might be made dependent on the magnitude of the dominant negative eigenvalue or made to match the length of the Newton or gradient step, or simply taken as unity. How to scale directions

of negative curvature is a question whose answer probably depends on the context of a particular algorithm which uses such directions.

2.4.3 Region of Trust. As we have seen, far from a solution the computed Newton step may be much too large, especially if the Jacobian or Hessian matrix is ill-conditioned, with the result that many function evaluations are wasted in the line search. One way of avoiding this problem is to define a region of trust, typically a ball of given radius about the current estimate for the solution, in which the linearization is considered adequate and therefore the Newton step is considered reliable. In some algorithms the trust radius might be simply a fixed, rather liberal estimate of the maximum length of a reasonable step. Any computed step exceeding this limit would either be scaled down or rejected entirely, so that absurdly large steps are avoided. A more sophisticated strategy is to try to estimate fairly closely the true radius of the trust region and adaptively adjust this estimate from iteration to iteration, reflecting the degree of success of the line searches based on previous estimates. Thus, if a restricted step gives descent, the restriction might be relaxed somewhat on the next iteration; if an unrestricted step fails, the trust radius would be reduced accordingly. Clearly many strategies of differing detail are possible.

2.4.4 Steplength Algorithms. The simple descent strategy of accepting any point which gives a function decrease, no matter how small, can result in very slow progress and does nothing to prevent successive steplengths from becoming arbitrarily small. In a sense, full benefit may not be

gained from a given search direction. This notion has led to a number of steplength algorithms which terminate only upon achieving a sufficiently large function decrease. For a survey of these see Section 8.3 of Ortega and Rheinboldt (1970). Important aspects of practical implementation are considered in Gill and Murray (1974b). Our discussion of this topic is in terms of problem UCON; analogous techniques apply to problems NLEQ and NLLS. There are two principal kinds of steplength algorithms.

1. Exact minimization. The greatest possible benefit to be gained from a given search direction would be given by the global minimum along that direction, as suggested by Cauchy in connection with the method of steepest descent. In practice, however, this proposal is quite unworkable since there are no algorithms guaranteed to find global minima, even for one-dimensional problems. Curry (1944) suggested the more tractable strategy of taking the first stationary point along the search direction--that is, the smallest positive root of the equation

$$g(x+\alpha p)^T p = 0 \quad ,$$

where p is the search direction. Although efficient and safe algorithms are available for reasonably well-behaved one-dimensional problems (see Brent (1973)), still a considerable number of function evaluations may be required. A further objection is that there is a kind of law of diminishing returns in that, beyond some point, additional accuracy in determining a minimum along a given search direction will have a negligible effect on the subsequent search direction. Thus, it would appear that

if a minimization strategy is to be followed in determining steplength, then a rather loose tolerance is called for, at least until very near the overall solution.

2. Sufficient decrease. In view of the potential inadequacy of simple descent and the possibly prohibitive cost of approximate one-dimensional minimization, steplength algorithms have been devised which fall between these two extremes, achieving a significant function decrease while guarding against steps that are too small. This work originated with Goldstein (1962) and has been further developed by Armijo (1966), Goldstein and Price (1967), Wolfe (1969, 1971), and others. Their proposals do not in fact constitute actual algorithms for generating trial points along the search direction but rather are more sophisticated termination criteria for conventional algorithms designed for simple descent or approximate minimization. The main idea is to predict how much function decrease can be expected along the given search direction, and stop only when some specified fraction of this decrease has been realized. A first order approximation to the change in the function f along the vector p is given by $\alpha g(x)^T p$. Thus, if η_1 is a fixed scalar, $0 < \eta_1 < 1$, the criterion for a sufficient decrease in f might be to require that α be such that

$$f(x) - f(x+\alpha p) \geq -\eta_1 \alpha g(x)^T p \quad .$$

Since this condition is automatically satisfied as $\alpha \to 0$, some additional requirement is needed to assure that steplengths do

not become too small. A simple way of bounding α away from zero is to require that, for fixed scalar η_2, $0 < \eta_2 < 1$,

$$g(x+\alpha p)^T p \geq \eta_2 g(x)^T p \quad .$$

If $\eta_1 \leq \eta_2$, then the two criteria are simultaneously satisfiable. In fact, if $\eta_1 < \eta_2$, then there is an interval of admissible steplengths, all of which give sufficient function decrease and are bounded away from zero. The normal procedure, then, is to allow the algorithm for generating trial points to continue until a trial point falls within this interval and is therefore accepted as the steplength. Criteria other than the above are possible; in particular, higher order information could be incorporated if available. Henceforth, the attainment of descent in a line search is meant to include the possibility of requiring satisfaction of some more exacting criterion of the kind discussed in this section, as well as simple descent (i.e., merely requiring a lower function value).

2.5 Hybrid Algorithms

The alternative strategies discussed in the last section have been combined in various ways to create hybrid algorithms which improve the robustness of the straightforward Newton method. Several of these algorithms are sketched in this section and a new one is proposed.

1. A tactic which is probably quite venerable and which appears in print, for example, in Goldstein and Price (1967) is to use the

Newton direction at each iteration if possible, but if trouble is encountered (rank-deficient or indefinite matrix, failure to attain descent, trust region exceeded, etc.), then switch to the negative gradient direction for the line search. This approach has a compelling simplicity about it and is difficult to improve upon if robustness (ability to survive adversity) is one's sole objective.

2. Gleyzal (1959) makes the interesting suggestion of searching simultaneously along both the Newton and gradient directions--that is, in the plane defined by points of the form $x + \alpha_1 s + \alpha_2 p$, where s is the Newton step and p is the negative gradient. He gives no indication, however, how such a two-dimensional search might be effectively carried out, and this idea would appear most impractical.

3. A more tractable way of combining the Newton and gradient directions is the Levenberg-Marquardt method discussed in the previous section. Note that $\mu = 0$ gives the usual Newton step while as $\mu \to \infty$ the direction of the computed step approaches that of the negative gradient. Furthermore, the length of the computed step approaches zero as $\mu \to \infty$ so that μ can serve as steplength parameter in attaining descent. Unfortunately, repeated solution of the linear system is required with each new value for μ. The effect of this drawback is lessened if the formulation (2.2) is used, since the orthogonal factorization of A^T can be computed once only and this factorization modified to account for the changes in $\mu^{\frac{1}{2}} I$. Another way to avoid repeated factorizations is to compute the

singular value decomposition of A^T (or eigenvalue decomposition of G) and observe that the singular values (eigenvalues) of the augmented matrix are just $(\sigma_i^2+\mu)^{\frac{1}{2}}$ (or $\gamma_i+\mu$). Either approach involves significantly more overhead than a simple steplength algorithm along a fixed direction, although in some cases this may be a minor consideration compared to the cost of function evaluations. Thus we are faced with the dilemma that it may be costly to allow μ to vary within an iteration but difficult to pick a suitable fixed μ for each iteration, especially since μ needs to be relatively large when far from a solution in order to attain descent, but must be small near a solution so as not to impede the convergence rate.

4. A way around this difficulty was proposed by Powell (1970a,b) for problems NLEQ and UCON, respectively, with his so-called "dogleg" algorithm. At the start of each iteration the Newton step s and scaled gradient step given by (2.3) (here denoted by p) are computed. Let r be the radius of the current region of trust. If $\|s\| \leq r$, then the Newton step is used as trail step. If $\|s\| > r$ and $\|p\| > r$, then the step $(r/\|p\|)p$ is used as trial step. If $\|s\| > r$ and $\|p\| \leq r$, then the point having norm r on the line $\{\alpha s + (1-\alpha)p : 0 \leq \alpha \leq 1\}$ is used as trial step. Whatever the choice, if the trial step fails to attain descent, then r is reduced by some fixed fraction and the process repeated. The effect of this strategy is to search for descent along the straight line between the endpoints of the vectors s and p, and if p is reached without descent having been attained,

the search continues along p toward the current point x. A larger value for r biases the trial step toward the Newton direction, while a smaller value for r biases the trial step toward the negative gradient direction. Thus, the intent is the same as that of Levenberg-Marquardt but without repeated solving of linear systems, since s and p are fixed throughout a given iteration. This idea has been further developed by Blue (1976), Welsch and Becker (1975), and Dennis and Mei (1975) for problems NLEQ, NLLS, and UCON, respectively.

5. Another algorithm in a similar vein is due to Jones (1970) who observed that the sequence of trial points generated by the Levenberg-Marquardt algorithm (as the parameter μ increases from zero to infinity) trace out a curve which begins at the end point of the Newton step and ends at the current base point x, where the curve is tangent to the gradient direction. In his algorithm Jones simulates this behavior by taking trial points which are weighted combinations of the Newton and gradient steps, with weights given by trigonometric formulas (the details of which are unimportant for our purposes) chosen so as to follow a spiral curve having the desired properties. Again only one linear system solution is required per iteration.

6. The hybrid algorithms discussed so far make use only of the Newton and negative gradient directions. Another possibility, at least for problem UCON, is to use a direction of negative curvature whenever the Hessian matrix is not positive definite. Such an algorithm is given by Fletcher and Freeman (1975) in which the

symmetric indefinite block-diagonal factorization of $G(x)$ is computed for each iteration in order to obtain either the Newton step or a direction of negative curvature, depending on whether the factorization shows $G(x)$ to be positive definite. A line search is then carried out along the direction computed. The case where $G(x)$ is singular requires special handling via either a direction of zero curvature or a pseudoinverse technique. As an alternative to Newton's method, directions of negative curvature have some advantages over steepest descent. First, it is possible to move away from saddlepoints, where the gradient and Newton step both vanish. Second, regions where the Hessian is positive definite are actively sought, since a negative eigenvalue tends to be eliminated each time a direction of negative curvature is followed until the function has an inflection point. However, the lack of convexity along a negative curvature direction may invalidate some line search algorithms and a more sophisticated strategy may be required.

7. Another algorithm which makes use of directions of negative curvature was proposed by McCormick (1976) and developed by Sorensen (1977). Here again Newton's method is used for each iteration in which the Hessian is positive definite. If, however, $G(x)$ has any negative eigenvalues, then a direction of negative curvature q is computed, and also an ordinary descent direction p (either the negative gradient or the Newton step given by $\bar{G}$). Then a line search takes place along a parameterized curve in the plane defined by the two vectors p and q. Thus, the trial

points are of the form $x_\alpha = x + \psi_1(\alpha)p + \psi_2(\alpha)q$, where the ψ_i are scalar valued functions. McCormick suggests the choices $\psi_1(\alpha) = \alpha^2$ and $\psi_2(\alpha) = \alpha$, which Sorensen justifies in the following way. Define

$$\Psi(\alpha) = f(x + \psi_1(\alpha)p + \psi_2(\alpha)q) \quad .$$

Then a Taylor series expansion in α about the origin gives

$$\Psi(\alpha) \approx \Psi(0) + \alpha\Psi'(0) + \frac{1}{2}\alpha^2\Psi''(0) \quad .$$

First, in order that $\Psi(0) = f(x)$, we require that $\psi_1(0) = \psi_2(0) = 0$. Now

$$\Psi'(0) = f(x)^T(\psi_1'(0)p + \psi_2'(0)q)$$

and

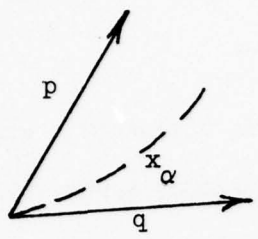
$$\Psi''(0) = g(x)^T(\psi_1''(0)p + \psi_2''(0)q)$$

$$+ (\psi_1'(0)p + \psi_2'(0)q)^T G(x)(\psi_1'(0)p + \psi_2'(0)q) \quad .$$

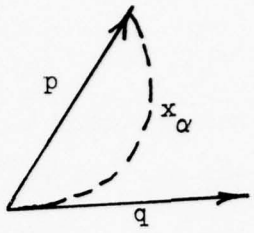
We want to ensure that $\Psi(\alpha) < \Psi(0)$ for small enough α. Since p^TGp is not necessarily negative, we require that $\psi_1'(0) = 0$. Consideration of the other terms yields the conditions $\psi_2'(0) \geq 0$, $\psi_1''(0) \geq 0$, and $\psi_2''(0) \geq 0$, with the potential decrease in Ψ greater if any of the inequalities are strict. The simplest functions having all of the above properties are given by $\psi_1(\alpha) = \alpha^2$ and $\psi_2(\alpha) = \alpha$. Both McCormick and Sorensen suggest steplength algorithms for this combined search which are

extensions of the termination criteria discussed previously for the standard case. Sorensen recommends systematic use of the symmetric indefinite block-diagonal factorization to determine if the Hessian is positive definite, perturb it if necessary to obtain $\bar{G}$, solve for the Newton step, and compute the negative curvature direction.

8. We now present an algorithm which is a synthesis of most of the ideas discussed thus far. First note that in the McCormick-Sorensen algorithm a full step along the descent direction p is never tried in the indefinite case since $\psi_2(\alpha) = 0$ only when $\alpha = 0$. Experience has shown, however, that the Newton step determined by $\bar{G}$ can be very fruitful. In order to give the pure Newton step a chance but still retain the overall character of the parameterized plane curve, we simply take $\psi_1(\alpha) = \alpha^2$ and $\psi_2(\alpha) = \alpha - \alpha^2$. Thus, we use trial points of the form $x_\alpha = x + \alpha^2 p + (\alpha-\alpha^2)q$, so that for $\alpha = 1$ the point $x+p$ is tried. Note that we still have $\psi_1(0) = \psi_2(0) = \psi_1'(0) = 0$, $\psi_2'(0) > 0$, and $\psi_1''(0) > 0$ as before. The only property lost is that now $\psi_2''(0) = -2$ (instead of $\psi_2''(0) = 0$), but the ultimate attainment of descent should not be inhibited since the second-order term involving ψ_2'' is dominated by first-order terms as $\alpha \to 0$ (i.e., the curve is still tangent to q at x). The two different curves are shown pictorially below.



$x_\alpha = x + \alpha^2 p + \alpha q$



$x_\alpha = x + \alpha^2 p + (\alpha - \alpha^2) q$

Now, however, the curve has taken on a familiar shape: if p is the Newton direction, and the negative gradient plays the role of q, then the curve simulates the Levenberg-Marquardt locus in much the same manner as the spiral curve in the algorithm of Jones discussed above. Thus there is a plane curve along which to search that is applicable to all three problems in all situations where a safe and efficient compromise is sought between a method which is slow but sure and one which is fast but risky. These considerations are summarized in the following algorithm schemata.

Algorithm 2.5 (for NLEQ or NLLS)

Initial data : x, r

Repeat until convergence tolerance met

1. Evaluate $c(x)$, $A(x)$
2. p := minimum (or basic) solution to $A^T p \cong -c$;
   $q := -(\|Ac\|^2 / \|A^T Ac\|^2) Ac$
3. Scale p and q, if necessary, to have maximum length r
4. Do line search on α for descent on $\varphi(x + \alpha^2 p + (\alpha - \alpha^2) q)$
5. Update x and r

Algorithm 2.6 (for UCON)

Initial data : x, r

Repeat until convergence tolerance met

1. Evaluate $g(x)$, $G(x)$
2. If G is positive definite then
    1. p := solution to $Gp = -g$
    2. $q := (g^T g / g^T G g) g$

    Else
    1. p := solution to $\bar{G}p = -g$
    2. q := direction of negative curvature
3. Scale p and q, if necessary, to have maximum length r
4. Do line search on α for descent on $f(x + \alpha^2 p + (\alpha - \alpha^2)q)$
5. Update x and r

Use of the estimated trust radius r in these algorithms is not absolutely essential. However it is helpful not only in restricting overall stepsize, but also in preventing undue bias in the weighted combination when one of the two vectors would otherwise be outsized.

What can go wrong in these two algorithms? The least-squares problem in Step 2 of Algorithm 2.5 is always solvable, even in the rank-deficient case. Of course, there may be some arbitrariness in estimating the rank of A, and the resulting solution is not necessarily useful, but at least the step can always be carried out. The denominator of the scale factor for q in Step 2 of Algorithm 2.5 vanishes only at a stationary point of φ (i.e., when $Ac = 0$). For Algorithm 2.6, in the positive definite case of Step 2 the solution of the linear system and the scale factor for the gradient are well defined; similarly for the perturbed linear system and

the direction of negative curvature in the indefinite case. The only anomalous situation is when G is semidefinite and singular. In this case the perturbed system may be used to obtain p, but it is not clear how best to define q. Assuming $g \neq 0$ (since otherwise the necessary conditions for a solution are satisfied) one could take q to be an alternate scaling of $-g$, or a direction of zero curvature (provided $g^T q < 0$), or let $q = 0$. If p and q computed in Step 2 of either algorithm are excessive in magnitude, this should be remedied in Step 3. Finally, since for small enough α the line search in Step 4 is essentially along the negative gradient, it should at least be able to attain simple descent. If failure occurs, however, Steps 1 through 5 might be retried with any approximate derivatives computed more accurately and with a smaller value for r.

Although the above discussion indicates that the individual steps of these algorithms can be carried out with a high probability of success, it is still possible for rounding or truncation errors or problem pathologies to cause failure. Because of the heuristic nature of some of the procedures involved and the use of finite precision arithmetic, it would be difficult to establish the most liberal conditions under which convergence could be proved. Nevertheless, the algorithms are reasonably immune to catastrophe and make a smooth, natural transition from the alternative procedures to Newton-like behavior in the vicinity of a solution.

2.6 Approximate Derivatives

As remarked earlier, it may be inconvenient or impractical to supply formulas for the derivatives required by Newton's method. Another drawback

to the standard formulation of the method is its computational overhead. The $O(n^2)$ scalar function evaluations needed to define the Jacobian or Hessian matrix plus the $O(n^3)$ arithmetic operations needed to factorize it at each iteration may be very expensive. Thus, a method which requires fewer function evaluations or arithmetic operations per iteration might have lower overall cost in solving a given problem, even if it were somewhat more slowly convergent. For example, for one-dimensional problems the secant method requires only one function evaluation per iteration while Newton's method requires evaluation of both the function and its derivative. Thus, since arithmetic overhead is negligible in this case for either method, the secant method may require less total work to attain a fixed accuracy, despite needing more iterations.

These considerations have led to a number of variants of Newton's method which lessen its programming inconvenience or computational overhead. Some of these variants are briefly reviewed in this section.

1. Finite differences. A common device for relieving the necessity to derive and code formulas for derivatives is to calculate approximate values for them by use of finite differences. For example,

$$g_i(x) \approx \frac{1}{h} \left(f(x+he_i) - f(x)\right)$$

or more accurately (but at a cost of an extra function evaluation)

$$g_i(x) \approx \frac{1}{2h} \left(f(x+he_i) - f(x-he_i)\right) \quad ,$$

where h is some small scalar steplength. Similarly, a symmetric approximate Hessian may be obtained by letting V be the matrix

whose i^{th} column is given by

$$v^{(i)}(x) = \frac{1}{h} (g(x+he_i) - g(x))$$

then taking $G = \frac{1}{2} (V+V^T)$. Note that such formulas usually do not save any computational work, since the extra function evaluations needed are likely to be as expensive as the equivalent derivative evaluations which they replace. Arithmetic overhead is also unaffected by this technique.

Surprisingly, perhaps, the theoretical convergence rate does not necessarily suffer with this kind of approximation, at least not if the steplength h is chosen carefully. In fact, quadratic convergence is still attainable in theory if h goes to zero rapidly enough as the iterates converge to the solution. See Dennis (1971) for a thorough discussion of this phenomenon. However, in practice on a computer there is a lower limit to how small h can be taken, and the question of how to choose h becomes rather delicate and depends on such factors as the machine precision and the relative local variation of the particular functions involved and how accurately they can be evaluated. Stewart (1967) proposes balancing the error due to cancellation in evaluating the finite difference expression with the truncation error of the formula. Murray (1972), p. 117, and others have advocated the use of a fixed value for h, typically about the square root of machine precision.

2. Periodic re-evaluation. A strategy which reduces both function

evaluations and arithmetic overhead is to evaluate and factorize the Jacobian or Hessian matrix only occasionally rather than on every iteration. Of course, the resulting scheme has a slower convergence rate, but this may be more than offset by the lower average cost per iteration. Net efficiency depends on the trade-off between frequency of derivative evaluation and overall convergence rate. This trade-off is quite problem-dependent and not easily determined in advance.

3. Neglected terms. A technique which reduces the complexity, if not the number, of derivative evaluations is to omit from the Jacobian or Hessian terms which vanish at the solution or are at least small enough to be considered negligible. The Gauss-Newton method for nonlinear least squares is an example of this idea. Local convergence of the resulting algorithm depends on the relative sizes of neglected terms; in particular, the convergence rate may still be quadratic if the missing terms really do vanish at the solution.

4. Secant updating. This is a family of methods variously known as quasi-Newton, variable metric, modification, or secant updating methods which originated with Davidon (1959), were further developed by Fletcher and Powell (1963), Broyden (1965), and many others, and are comprehensively surveyed in Dennis and Moré (1977). These methods require only function values for problem NLEQ and function and gradient values for problem UCON, and involve only $O(n^2)$ arithmetic operations per iteration. An approximation to the Jacobian or Hessian is built up sequentially from iteration to iteration based on the change in function or gradient values

between successive iterates. Because of the simple form of the rank-one or rank-two change in the matrix at each iteration, the factorization of the previous matrix can be updated at a cost of $O(n^2)$ operations, as opposed to the $O(n^3)$ cost if the new matrix were factorized from scratch (see Gill, Golub, Murray and Saunders (1974)). The convergence rate of these methods is still superlinear, even though the approximate Jacobian or Hessian does not necessarily converge to the true Jacobian or Hessian as the iterates converge to a solution. An additional interesting feature of these methods is that the initial approximate Jacobian or Hessian is often taken to be the identity matrix, so that in the early stages the algorithm acts like the method of steepest descent but becomes more Newton-like as the approximation improves.

CHAPTER 3. CONSTRAINED PROBLEMS

3.1 Newton's Method for Problem ECON

We now consider optimization problems with constraints. Our approach is to cast these problems into a framework which allows the techniques developed in Chapter 2 to be applied. The equality constrained (ECON) and inequality constrained (ICON) cases are treated separately at first. The main practical difference is that with inequalities it is not known at the outset which constraints are active at a solution. A survey of previous methods for constrained problems is postponed until Section 3.4, since it will be convenient to refer to machinery to be developed presently.

We begin with problem ECON, which, to recall notation, is to find

$$\min_{x} \{f(x) : c(x) = 0\} \quad ,$$

where $f:\mathbb{R}^n \to \mathbb{R}$ and $c:\mathbb{R}^n \to \mathbb{R}^m$ with $m \leq n$. The classical Lagrangian function $L:\mathbb{R}^{n+m} \to \mathbb{R}$ for this problem is defined by

$$L(x,\lambda) = f(x) + \lambda^T c(x)$$

so that its gradient and Hessian are given by

$$w(x,\lambda) = g(x) + A(x)\lambda$$

and

$$W(x,\lambda) = G(x) + \sum_{i=1}^{m} \lambda_i G_i(x) \quad .$$

Under appropriate regularity assumptions (see Theorem 1.1 in Section 1.4), the first-order necessary condition for optimality is that

$$\begin{pmatrix} w(x,\lambda) \\ c(x) \end{pmatrix} = 0 \quad . \tag{3.1}$$

This immediately suggests treating (3.1) as a system of nonlinear equations of order $n+m$ and using Newton's method to solve for (x^*,λ^*). Given an estimate (x,λ) for a solution, the linear system to be solved for the Newton correction (s,δ) is

$$\begin{pmatrix} B & A \\ A^T & 0 \end{pmatrix}\begin{pmatrix} s \\ \delta \end{pmatrix} = -\begin{pmatrix} w \\ c \end{pmatrix}, \tag{3.2}$$

where B denotes the symmetric matrix playing the role of $W(x,\lambda)$, the Hessian of the Lagrangian function. (The use of a different symbol is meant to emphasize the possibility that B may be only an approximation to the true Hessian.) Then the new estimate for the solution is $(x+s,\lambda+\delta)$. It is important to note that while the matrix of system (3.2) is symmetric, it is never positive definite--in keeping with the saddlepoint nature of solutions to problem ECON--so that pivoting is generally required in solving the system directly.

Such an approach has the usual advantages and disadvantages expected of Newton's method, such as having rapid--but rather localized--convergence. In addition there are several other difficulties in this context:

a. Increasing the dimension of the problem to $n+m$ seems highly undesirable in view of the work involved in solving so large a linear system.

b. A starting guess is now needed for the Lagrange multiplier vector λ as well as for the solution x.

c. Since the objectives of finding a lower value of the function f and of reducing infeasibility may conflict, it is not clear how to define or monitor progress toward a solution.

d. It is not obvious how to tell whether a solution to (3.1), when found, is in fact a constrained minimum, since the space on which B should be positive definite is not explicitly available.

We now investigate ways to overcome these deficiencies in order to develop an efficient and robust algorithm.

3.1.1 Solving the Linear System. The presence of the zero block in the lower-right corner of (3.2) allows the system to be uncoupled into separate, smaller systems for s and δ. This approach is taken by Kelley, et al (1966), Bard and Greenstadt (1969), Golub and Saunders (1969), and Tapia (1974), among others.

The upper row of (3.2) gives the equation

$$Bs = -A\delta - w \quad . \tag{3.3}$$

If B is nonsingular, premultiplying both sides by A^TB^{-1} and rearranging yields

$$A^TB^{-1}A\delta = -A^Ts - A^TB^{-1}w \quad .$$

Noting from the lower row of (3.2) that $A^Ts = -c$, we finally obtain

$$A^TB^{-1}A\delta = c - A^TB^{-1}w \quad . \tag{3.4}$$

Equation (3.4) is independent of s, so that, if the matrix $A^TB^{-1}A$ is nonsingular, system (3.4) can be solved for δ. Once δ is known, s can then be computed from the linear system (3.3). Although solving the symmetric (but in general indefinite) systems (3.4) and (3.3) appears to represent a considerable computational savings over the direct solution of

(3.2), this gain is more than offset by the heavy cost of forming the matrix $A^TB^{-1}A$ needed in (3.4). (See Appendix for a comparison of operation counts.) Furthermore, this approach suffers from the same potential numerical instability as the normal equations for linear least squares in that explicit formation of the matrix $A^TB^{-1}A$ worsens the conditioning of the problem and can cause substantial loss of accuracy. Thus, this method is inferior, both in efficiency and stability, to simply solving the system (3.2) directly by the block-diagonal-pivoting algorithm, and therefore has little to recommend it.

An alternative approach to the solution of (3.2) is to be found in Orden (1964). For simplicity of presentation assume that A has rank m and let Z be an $n \times (n-m)$ matrix whose columns form a basis for the null space of A. Then the vector s can be written as a sum

$$s = Au + Zv$$

of range- and null-space components. Since $A^TZ = 0$, the lower row of (3.2) gives the equation

$$A^Ts = A^T(Au+Zv) = A^TAu = -c \quad , \tag{3.5}$$

which can be solved for u. Similarly, the upper row of (3.2) gives the equation

$$Bs = B(Au+Zv) = -w - A\delta \quad .$$

Premultiplying by Z^T and rearranging leads to the equation

$$Z^TBZv = -Z^T(w+BAu) \quad , \tag{3.6}$$

which can be solved for v. In this way s is determined without having computed δ. In order to obtain δ, note that δ must satisfy exactly the overdetermined system

$$A\delta = -w - Bs \quad , \tag{3.7}$$

which may be solved by least squares or by selecting any m linearly independent rows and solving the resulting square problem. Orden suggested the use of Gauss-Jordan reduction on A^T to compute the matrix Z. Suggestions similar in spirit are made by Wolfe (1967) and McCormick (1970). (See Appendix for a discussion of all three methods.)

As in the linear least-squares problem, however, an approach which is superior numerically is provided by orthogonalization (see, for example, Gill and Murray (1974a), pp. 61-63). If Y and Z are chosen as in (1.1a) with Q as in (1.1), then Y and Z are orthogonal bases for the range and null spaces of A. Thus s may be written as

$$s = s_R + s_N = Yp_R + Zp_N \quad ,$$

where the obvious notation is used for range- and null-space components. Then the analogues of (3.5) and (3.6) are

$$A^T s = A^T(s_R+s_N) = A^T(Yp_R+Zp_N) = R^T p_R = -c \tag{3.8}$$

and

$$Z^T BZp_N = -Z^T(g+Bs_R) \quad . \tag{3.9}$$

(In deriving (3.9) we have used the fact that $Z^T w = Z^T(g+A\lambda) = Z^T g$.) Also, premultiplying (3.7) by Y^T gives

$$Y^T A\delta = R\delta = -Y^T(w+Bs) \quad . \tag{3.10}$$

Note that systems (3.8) and (3.10) are triangular and therefore particularly easy to solve. It can be shown that this choice of Z minimizes the condition number of Z^TBZ (see Gill and Murray (1974a), p. 61). Use of (3.8)-(3.10) requires the orthogonal factorization of the $n \times m$ matrix A, formation of the matrix Z^TBZ, and solution of the symmetric system (3.9) of order $n-m$ and the triangular systems (3.8) and (3.10) of order m. The relative efficiency of carrying out these steps depends on the ratio of m to n, but generally this method is roughly comparable in efficiency to the use of (3.3)-(3.4) or direct solution of (3.2) by the block-diagonal-pivoting algorithm (see Appendix for a comparison of operation counts). However, in addition to numerical stability the orthogonalization approach has the advantage of providing important information which is not directly available with the other two methods. In particular we shall make vital use of the matrix Z^TBZ and the range- and null-space components of s.

The matrix Z^TBZ may be formed in a variety of ways, depending on whether the matrices B and Z are explicitly available. Suppose that B is explicitly available, say, from analytic second derivatives or a quasi-Newton approximation. Then, if Z is also explicitly available, Z^TBZ may be computed by ordinary matrix multiplication. If, on the other hand, Q is retained as a sequence of m individual Householder transformations, then these may be applied successively to B as congruence transformations. The latter is more efficient if $m \ll n$, while an explicitly formed Z requires less total work if $m \approx n$ (see Appendix for precise comparisons). If B is not explicitly available and second

derivatives are to be approximated by finite differences, then a saving in first derivative evaluations, as well as one matrix multiplication, results from forming BZ directly by finite differencing $w(x,\lambda)$ along the $n-m$ columns of Z rather than the n unit vectors e_i. Vectors such as Bs needed in (3.9) and (3.10) may be similarly formed by a finite difference of $w(x,\lambda)$ along s.

3.1.2 Estimating Lagrange Multipliers. For a given problem a reasonable initial estimate for the solution x may be available, but this is unlikely to be true for the corresponding starting value for the Lagrange multiplier λ. Therefore it is desirable to estimate this vector using only x and the corresponding problem data. (See Gill and Murray (1977) for a thorough discussion of this topic.) A standard technique is based on the necessary condition $w(x^*,\lambda^*) = 0$ for a constrained minimum (see Theorem 1.1), which may be written as

$$A(x^*)\lambda^* = -g(x^*) \quad .$$

At an arbitrary point x such a relationship is not, in general, exactly satisfiable. However, λ may be estimated by solving in the least-squares sense the overdetermined system

$$A\lambda \cong -g \quad . \tag{3.11}$$

Geometrically, this amounts to projecting the negative gradient of the objective function f into the linear span of the constraint normals.

Note that with λ given by (3.11), w becomes simply the residual vector for this least-squares problem. Therefore w lies in the null space

of A and the system (3.7) is then equivalent to the least-squares problem

$$A\delta \cong -Bs \quad . \tag{3.12}$$

In this way λ given by (3.11) may be regarded as a first-order estimate for the Lagrange multiplier and δ given by (3.12) as a second-order correction term. (Gill and Murray (1974a), p. 41, make these notions precise.) The least-squares problems (3.11)-(3.12) are best solved as in Section 1.5.1 by means of the orthogonal factorization of A, which is needed for (3.8)-(3.9) in any case.

First, the triangular system

$$Y^TA\lambda = R\lambda = -Y^Tg \tag{3.13}$$

is solved for λ, which is then used in defining those quantities which depend on λ such as w and B (or Z^TBZ). Now $w = g + A\lambda$, but for storage economy in a computer program A may be destroyed before λ is known. Thus, it is computationally convenient to note that

$$w = ZZ^Tg \quad ,$$

since ZZ^T is the orthogonal projector onto the null space of A (see Section 1.5.1).

Although the above procedure for estimating λ was stated as a means of getting started on the first iteration of Newton's method, it can in fact be applied on every iteration. After x is updated then a new value for λ can be computed from (3.11) rather than using the updated value $\lambda+\delta$ from the previous iteration. The heuristic motivation for doing this

is that a first-order estimate for the Lagrange multiplier vector at the new point is likely to be better than a second-order estimate at the old point (see Gill and Murray (1977), p. 13). The effect of this startegy is to make the iteration scheme independent of λ, and this implies that Newton's method--in the strict sense of (3.2)--is no longer being used. However, the resulting algorithm is still locally convergent with second-order asymptotic convergence rate (in the exact derivative case) as shown in the following theorem.

Theorem 3.1. Let x^*, with associated Lagrange multiplier λ^*, be a regular solution to problem ECON satisfying the sufficient conditions of Theorem 1.1. Let $\lambda: \mathbb{R}^n \to \mathbb{R}^m$ be a function which is differentiable in a neighborhood of x^* and such that $\lambda(x^*) = \lambda^*$. (In particular, λ given by (3.11) is such a function.) Let s be given by (3.2) with all derivatives exact. Then the iterative process $x := x+s$ is locally convergent to x^* with second-order asymptotic convergence rate.

Proof. If the iteration operator is written as $S(x) = x+s$, then by Theorem 10.1.6 of Ortega and Rheinboldt (1970), p. 303, it suffices to show that $S'(x^*) = 0$. For purposes of proof we may use the mathematically equivalent expressions (3.5) and (3.6) to determine s. Thus,

$$\begin{aligned} S(x) &= x + s = x + Au + Zv \\ &= x - A(A^TA)^{-1}c - Z(Z^TBZ)^{-1}Z^T(w+BAu) \\ &= x - A(A^TA)^{-1}c - Z(Z^TBZ)^{-1}Z^T(w-BA(A^TA)^{-1}c) \quad , \end{aligned}$$

where all functions are evaluated at $(x,\lambda(x))$. Using the fact that $c(x^*) = 0$ and $w(x^*,\lambda^*) = 0$, straightforward differentiation at x^* shows

that

$$S'(x^*) = I - A(A^TA)^{-1}A^T - Z(Z^TBZ)^{-1}Z^T(B-BA(A^TA)^{-1}A^T)$$

$$= [I-Z(Z^TBZ)^{-1}Z^TB][I-A(A^TA)^{-1}A^T]$$

$$= [I-Z(Z^TBZ)^{-1}Z^TB]ZZ^T$$

$$= ZZ^T - ZZ^T = 0 \quad .$$

Remarks. In this derivation we have used the fact that $I - A(A^TA)^{-1}A^T = ZZ^T$, which is an easily verified property of orthogonal projectors (see, for example, Householder (1964), p. 8). The existence of the indicated matrix inverses in a neighborhood of x^* follows from the assumptions that $A(x^*)$ has full rank and that $B(x^*,\lambda^*)$ is positive definite on M_{x^*}.

3.1.3 Minimization vs. Feasibility. In the spirit of Newton's method for general nonlinear systems it might appear that all that is needed in order to monitor progress toward a solution of ECON is a norm on $\mathbb{R}^{n+m}$ with which to measure the size of the right-hand side of (3.2). However, any such norm represents an implicit weighting of the two distinct components w and c, and of the corresponding objectives of finding a stationary point of the Lagrangian function and attaining feasibility. Through its influence on the descent criterion of the line search, this weighting may markedly affect the path taken toward a solution and may even mean the difference between success and failure. Furthermore, merely finding a zero of (3.1) does nothing to avoid constrained stationary points which are not minima. The latter difficulty might be circumvented by minimizing $L(x,\lambda)$ in the line search rather than $\|w(x,\lambda)\|$. Unfortunately, however, the Lagrangian

function does not necessarily have a minimum nor does it necessarily decrease along the Newton direction. In some circumstances these deficiencies may be remedied by augmenting the Lagrangian function with an additional term which penalizes constraint violation (this approach will be discussed in Section 3.4.2). However, such methods require a possibly delicate choice of a parameter which itself represents a somewhat arbitrary relative weighting of minimization vs. feasibility. Moreover, a suitable value for the parameter may not exist. Since avoiding such parameters is one of our design goals, we take a different approach.

Assuming A has full rank, the range-space component of the Newton step, $s_R = Yp_R$, where p_R is given by (3.8), is a descent direction for $\varphi(x) = \frac{1}{2}\|c(x)\|^2$, since

$$\nabla\varphi(x)^T s_R = (Ac)^T Yp_R = c^T A^T Yp_R = c^T R^T p_R = -c^T c \leq 0 \quad .$$

At the same time, since w lies in the null space of A, local movement along s_R tends to have a relatively small effect on the value of $L(x,\lambda)$. This suggests that a line search be carried out along s_R with the sole objective of attaining descent on $\varphi(x)$.

The analogous situation with regard to the null-space component of the Newton step, $s_N = Zp_N$, where p_N is given by (3.9), is less clear. Although s_N often is a descent direction for $L(x,\lambda)$, especially when s_R is small (e.g., near a solution), s_N may fail to be a descent direction for $L(x,\lambda)$. However, if $s_N^{(1)} = Zp_N^{(1)}$, with $p_N^{(1)}$ given by the system

$$Z^T B Z p_N^{(1)} = -Z^T g \quad ,$$

then $s_N^{(1)}$ is guaranteed to be a descent direction for both $L(x,\lambda)$ and $f(x)$, since

$$w^T s_N^{(1)} = g^T s_N^{(1)} = g^T Z p_N^{(1)} = -g^T Z (Z^T B Z)^{-1} Z^T g \leq 0 .$$

On the other hand, in order to retain true Newton-like behavior near a solution, it is desirable to use s_N as originally defined if possible. Thus, in practice $s_N^{(1)}$ should be computed and also $s_N^{(2)} = Z p_N^{(2)}$, where $p_N^{(2)}$ is the solution to the linear system

$$Z^T B Z p_N^{(2)} = -Z^T B s_R .$$

(Of course, the matrix need be factorized only once in order to solve both systems.) Then let $s_N := s_N^{(1)} + s_N^{(2)}$ if this gives a descent direction (i.e., $w^T s_N \leq 0$), otherwise let $s_N := s_N^{(1)}$. Sometimes it may also be advisable to suppress $s_N^{(2)}$ on other grounds--for example, when $Z^T B Z$ is indefinite or if $\|s_R\|$ or $\|s_N\|$ exceeds the trust radius. In any case, we henceforth assume that s_N is chosen to be a descent direction for $L(x,\lambda)$ and $f(x)$.

Since the null space of A is in a sense parallel to the constraint surface, local movement along s_N tends to have a relatively small effect on $\varphi(x) = \|c(x)\|^2$. This suggests that a line search be carried out along s_N with the sole objective of attaining descent on $L(x,\lambda)$ or $f(x)$ (since s_N is a descent direction for both, either may be used as merit function). Thus, the conflict between the dual objectives of minimizing the function f and attaining feasibility is resolved by conducting separate line searches in the range and null spaces of A. The total step taken is the sum of the two individual steps resulting from these line searches. The composite step for a given iteration is not guaranteed to produce progress toward a

solution, however measured, even when both line searches are successful in attaining descent for their respective merit functions. The general trend, though, is usually favorable. Moreover, near a solution full steps tend to be taken in each search direction, leading to Newton-like asymptotic behavior.

Another advantage of this two-part line search is that it makes available the techniques developed in Chapter 2 for improving the robustness of the algorithm far from a solution. In particular, s_R may be regarded as a Newton step in the range space of A, and Ac as the corresponding gradient of $\varphi(x)$, so that strategies such as Algorithm 2.5 may be used in seeking descent on $\varphi(x)$. Similarly, s_N may be regarded as a Newton-like step in the null space and $w = ZZ^Tg$ as the corresponding gradient, so that strategies such as Algorithm 2.6 may be used in seeking descent on $L(x,\lambda)$ or $f(x)$. As in the unconstrained case, if the matrix Z^TBZ is indefinite, a direction of negative curvature $q_N \in \mathbb{R}^{n-m}$ is computed and the transformed vector $q = Zq_N$ used in the line search. Note that this strategy allows the algorithm to steer clear of or descend from constrained stationary points which are not minima.

3.1.4 Checking a Solution. As observed by Orden (1964), the second-order necessary condition for optimality implies that at a solution to problem ECON the matrix Z^TBZ of (3.6) or (3.9) should be positive semi-definite, since the columns of Z form a basis for the tangent space M_{x^*} to the constraint manifold (see Theorem 1.1). This fact gives an effective way of computationally checking a constrained stationary point (i.e., a solution to (3.1)) to determine whether it is a solution to problem ECON.

The inertia of the matrix Z^TBZ is a simple byproduct of its block-diagonal factorization (see Section 1.5.2), which is needed for solving system (3.9). Of course, the issue may be clouded somewhat by rounding error or if B is only an approximation to the true Hessian of the Lagrangian function. Nevertheless, within these limitations there is a theoretically justifiable basis for accepting or rejecting a candidate for a solution.

Another implication of the positive definiteness of Z^TBZ at a solution is that is is reasonable to perturb Z^TBZ, when it is indefinite, in order to replace it by a positive definite matrix as in Section 2.4.1, since such a perturbation should only be needed away from a solution. Because the matrix B is not necessarily positive definite at a solution, to perturb B in this manner in order to solve (3.3) or (3.4) runs the risk of slowed convergence or outright failure, since the algorithm would no longer simulate Newton's method near a solution.

3.2 An Algorithm for Problem ECON

The use of (3.8) and (3.9) in solving for the range- and null-space components of the Newton step s and (3.13) in solving for the Lagrange multiplier vector λ are incorporated into the following algorithm for problem ECON. Its main novel feature is the use of separate range- and null-space line searches in a Newton-like mode, which in turn allows direct use of the alternative procedures and hybrid strategies developed in Chapter 2 in order to improve the robustness of the algorithm.

Algorithm 3.1 (for ECON)

Initial data : x

Repeat until convergence tolerance met

1. Evaluate $g(x)$, $c(x)$, $A(x)$

2. Compute orthogonal matrix $Q^T = (Y,Z)$ such that $QA = \begin{pmatrix} R \\ O \end{pmatrix}$ with R upper triangular as in (1.1)

3. Solve triangular system $R\lambda = -Y^T g$

4. $w := ZZ^T g$

5. Solve triangular system $R^T p_R = -c$

6. $s_R := Yp_R$

7. Do line search on α_R for descent on $\varphi(x+\alpha_R s_R) = \frac{1}{2} \|c(x+\alpha_R s_R)\|^2$

8. Form Bs_R

9. Form $Z^T BZ$

10. Solve symmetric system $Z^T BZ p_N = -Z^T(g+Bs_R)$ by block-diagonal-pivoting algorithm

11. $s_N := Zp_N$

12. Do line search on α_N for descent on $L(x+\alpha_N s_N, \lambda)$ or $f(x+\alpha_N s_N)$

13. $x := x + \alpha_R s_R + \alpha_N s_N$

As discussed in Section 1.5.1, column pivoting should be done in Step 2 in order to detect and properly handle rank deficiency. The resulting triangular matrix R should then be nonsingular and reasonably well conditioned, although its order--hence, the number of columns of Y--may be less than m. In the latter case, for any column dropped from A due to linear dependence the corresponding component of c should be deleted from the linear system of Step 5. If second derivatives are approximated by finite difference, then Bs_R and $Z^T BZ$ in Steps

is available is that the resulting algorithm no longer has simple Newton-like behavior, even near a solution. Thus, rapid asymptotic convergence cannot be expected necessarily. For these reasons such variants of Algorithm 3.1 will not be discussed further here (however, see Section 3.4.3).

For simplicity Algorithm 3.1 was presented solely as a Newton-like algorithm. However, it should be clear how to incorporate the gradient Ac into the range-space search and the gradient w (or a direction of negative curvature when Z^TBZ is indefinite) into the null-space search, as well as the use of a trust radius, in order to implement hybrid strategies such as Algorithms 2.5 and 2.6 in attaining descent in the respective line searches. Note that Algorithm 3.1 would be forced to terminate in failure on a given iteration only if both line searches failed to attain descent for their respective merit functions.

By skipping unnecessary steps, a properly coded implementation of Algorithm 3.1 should be able to solve problems NLEQ and UCON. Thus, Step 2 followed by Steps 5 through 7 together solve the nonlinear system $c(x) = 0$. By taking $\lambda = 0$, $s_R = 0$, and $Z = I$, Steps 9 through 12 accomplish the unconstrained minimization of the function f.

3.3 An Algorithm for Problem ICON

In this section the algorithm of the previous section is extended to handle optimization problems having inequality constraints. Thus we consider the problem ICON:

$$\min_x \{f(x) : c(x) \leq 0\} \quad ,$$

where $f:\mathbb{R}^n \to \mathbb{R}$ and $c:\mathbb{R}^n \to \mathbb{R}^m$. The presence of inequalities brings about a considerable increase in difficulty over the equality constrained case. Much of this additional difficulty is due to the problem of determining which of the constraints are active at a solution, a problem with a combinatorial nature not present in the equality constrained case.

Newton's method, in the usual sense, is not directly applicable to inequality constrained problems. Therefore, a number of methods have been proposed which convert problem ICON into an equality constrained problem or an unconstrained problem. These methods include the use of slack variables (Klein (1955)), transformations of variables (Box (1966)), and penalty functions (Courant (1943), Carroll (1961), Fiacco and McCormick (1968)). Though theoretically valid, in actual implementation these methods unfortunately may have harmful effects of great practical significance. The possible effects include the introduction of spurious minima or stationary points, an increase in the dimension of the problem, dependence on choice of parameters, and numerical instability. Rockafellar (1973) introduced a modified Lagrangian function designed to explicitly allow inequalities, but it has the disadvantage of not being differentiable at some points, so that the applicability of Newton-like methods is questionable.

Still another approach is provided by <u>active set</u> strategies, which amount to guessing an "active set" of constraints believed to be active at the solution (often taken to be those currently binding or violated), carrying out one step of an iterative algorithm for--or perhaps completely solving--the resulting equality constrained problem, then revising the active constraint set at the new point. One potential difficulty with such strategies is the possibility of <u>jamming</u>, a phenomenon in which a given constraint is repeatedly added to, then deleted from, the active set. See

8-9 may be approximated directly as discussed in Section 3.1.1. In this case the matrix Z must be formed explicitly rather than implicitly represented by the sequence of Householder transformations used in triangularizing A. The convergence test could come immediately after Step 10, using the inertia of Z^TBZ that is a byproduct of its block-diagonal factorization (see Section 1.5.2). Thus, if the norms of w and c are within a specified small tolerance and Z^TBZ is positive semidefinite, convergence to a solution is declared (see Theorem 1.1).

Although the range-space step $\alpha_R s_R$ is determined as early as Step 7, no updating of x is done until Step 13. As a sort of "Gauss-Seidel" strategy of always using the latest information, it might appear that the update $x := x + \alpha_R s_R$ should be done immediately after Step 7, or at least before the subsequent line search in Step 12. If x were updated immediately after Step 7, however, then the problem data (in particular, Z) already computed and needed for Steps 9 and 10 would either be obsolete or require re-evaluation. (This objection is mitigated by the fact that $Z^T(g(x)+Bs_R)$ approximates $Z^Tw(x+s_R,\lambda)$.) Another possibility along this line would be to repeat Steps 1 through 7--updating x after each line search--until $c(x) \approx 0$ within some tolerance, then proceed to Steps 8 through 12. The repetition of Steps 1 through 7 in order to obtain an approximate feasible point can sometimes be helpful initially when the starting point is very far from the feasible region. If this policy is continued, however, the algorithm will tend to generate small steps which closely follow the constrait manifold, resulting in very slow progress toward a solution. Another disadvantage with updating x as soon as s_R

is available is that the resulting algorithm no longer has simple Newton-like behavior, even near a solution. Thus, rapid asymptotic convergence cannot be expected necessarily. For these reasons such variants of Algorithm 3.1 will not be discussed further here (however, see Section 3.4.3).

For simplicity Algorithm 3.1 was presented solely as a Newton-like algorithm. However, it should be clear how to incorporate the gradient Ac into the range-space search and the gradient w (or a direction of negative curvature when Z^TBZ is indefinite) into the null-space search, as well as the use of a trust radius, in order to implement hybrid strategies such as Algorithms 2.5 and 2.6 in attaining descent in the respective line searches. Note that Algorithm 3.1 would be forced to terminate in failure on a given iteration only if both line searches failed to attain descent for their respective merit functions.

By skipping unnecessary steps, a properly coded implementation of Algorithm 3.1 should be able to solve problems NLEQ and UCON. Thus, Step 2 followed by Steps 5 through 7 together solve the nonlinear system $c(x) = 0$. By taking $\lambda = 0$, $s_R = 0$, and $Z = I$, Steps 9 through 12 accomplish the unconstrained minimization of the function f.

3.3 An Algorithm for Problem ICON

In this section the algorithm of the previous section is extended to handle optimization problems having inequality constraints. Thus we consider the problem ICON:

$$\min_x \{f(x) : c(x) \leq 0\} \ ,$$

where $f:\mathbb{R}^n \to \mathbb{R}$ and $c:\mathbb{R}^n \to \mathbb{R}^m$. The presence of inequalities brings about a considerable increase in difficulty over the equality constrained case. Much of this additional difficulty is due to the problem of determining which of the constraints are active at a solution, a problem with a combinatorial nature not present in the equality constrained case.

Newton's method, in the usual sense, is not directly applicable to inequality constrained problems. Therefore, a number of methods have been proposed which convert problem ICON into an equality constrained problem or an unconstrained problem. These methods include the use of slack variables (Klein (1955)), transformations of variables (Box (1966)), and penalty functions (Courant (1943), Carroll (1961), Fiacco and McCormick (1968)). Though theoretically valid, in actual implementation these methods unfortunately may have harmful effects of great practical significance. The possible effects include the introduction of spurious minima or stationary points, an increase in the dimension of the problem, dependence on choice of parameters, and numerical instability. Rockafellar (1973) introduced a modified Lagrangian function designed to explicitly allow inequalities, but it has the disadvantage of not being differentiable at some points, so that the applicability of Newton-like methods is questionable.

Still another approach is provided by <u>active set</u> strategies, which amount to guessing an "active set" of constraints believed to be active at the solution (often taken to be those currently binding or violated), carrying out one step of an iterative algorithm for--or perhaps completely solving--the resulting equality constrained problem, then revising the active constraint set at the new point. One potential difficulty with such strategies is the possibility of <u>jamming</u>, a phenomenon in which a given constraint is repeatedly added to, then deleted from, the active set. See

Wolfe (1972) or Zangwill (1969), pp. 270-280, for discussion and examples of this problem. Unless care is taken to avoid it (see, for example, the rules formulated by Zoutendijk (1960), pp. 72-74), such behavior could lead to nonconvergence or convergence to a nonoptimum point.

The approach we take does not fall strictly within any of the above categories, although it has elements in common with modified Lagrangian, penalty function, and active set strategies. A primary concern is that the treatment of inequalities fit naturally into the framework already established, so that, for example, we may use the same linear algebra and line search techniques as in previous algorithms. Thus, in order to apply Newton-like methods, we insist on differentiability. We also wish to maintain the same efficiency and numerical stability as before, while avoiding such potential difficulties as spurious solutions, dependence on parameters, increased dimension, and jamming. Our point of departure in pursuit of these objectives is the following theorem of Mangasarian (1976).

Theorem 3.2. Define the function $d:\mathbb{R}^{n+m} \to \mathbb{R}^m$ componentwise by letting

$$d_i(x,\lambda) = (c_i(x)+\lambda_i)^2 + c_i(x)|c_i(x)| - \lambda_i|\lambda_i| , \qquad i = 1,\ldots,m ,$$

and let the function $F:\mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$ be given by

$$F(x,\lambda) = \begin{pmatrix} w(x,\lambda) \\ d(x,\lambda) \end{pmatrix} .$$

Then $F(x,\lambda) = 0$ if, and only if, $w(x,\lambda) = 0$, $c(x) \leq 0$, $\lambda \geq 0$, and $\lambda^T c(x) = 0$. (Note from Theorem 1.1 that these are just the first-order necessary conditions for a solution to problem ICON.) If f and c are

twice differentiable, then F is differentiable. Moreover, if x^* is a regular solution to problem ICON, with associated Lagrange multiplier vector λ^*, satisfying the second-order sufficient condition of Theorem 1.1 (i.e., $W(x^*,\lambda^*)$ is positive definite on the tangent space M_{x^*} to the active constraint manifold) and strict complementarity holds (i.e., $\lambda^* = 0 \iff c(x^*) \neq 0$), then $F'(x^*,\lambda^*)$ is nonsingular.

Proof. See Mangasarian (1976), pp. 174-175.

Remarks. Although similar in spirit and motivation to the modified Lagrangian method of Rockafellar (1973), this theorem does not, strictly speaking, represent a modified Lagrangian approach, since the function F is not the gradient of any functional on $\mathbb{R}^{n+m}$. The assumption of strict complementarity is quite reasonable in practice since a constraint which is active at a solution, but has a zero Lagrange multiplier, is redundant in the sense that the same point would still be a local minimum without that particular constraint.

The importance of this theorem is that it converts problem ICON into a system of nonlinear equations, $F(x,\lambda) = 0$, and does so without introducing any parameters or slack variables. Furthermore, the system is smooth, so that it can be solved by Newton-like methods, and has (under mild conditions) a nonsingular Jacobian at a solution, so that rapid asymptotic convergence can be expected.

Recalling that for scalars $\frac{d}{d\xi}(\xi \cdot |\xi|) = 2 \cdot |\xi|$, straightforward differentiation shows the Jacobian of F to be given by

$$F'(x,\lambda) = \begin{bmatrix} W(x,\lambda) & A(x) \\ \Lambda(x,\lambda)A(x)^T & D(x,\lambda) \end{bmatrix},$$

where $\Lambda(x,\lambda)$ and $D(x,\lambda)$ are diagonal matrices with diagonal elements given by

$$\{\Lambda(x,\lambda)\}_{ii} = 2(c_i(x) + |c_i(x)| + \lambda_i)$$

and

$$\{D(x,\lambda)\}_{ii} = 2(c_i(x) + \lambda_i - |\lambda_i|) \quad .$$

In evaluating F' at a solution, let us assume that the component constraint functions are numbered so that the first $\hat{m}$, say, are active at x^* and the remaining $m-\hat{m}$ are not. As usual, the "^" notation over any vector or matrix indicates that only active constraints are included; similarly, we introduce a dual symbol "v" to indicate that only components or elements corresponding to inactive constraints are included. With these conventions, at a solution x^* to problem ICON, with corresponding Lagrange multiplier vector λ^*, the Jacobian of F is

$$F'(x^*,\lambda^*) = \begin{bmatrix} W(x^*,\lambda^*) & \hat{A}(x^*) & \overset{v}{A}(x^*) \\ \hat{\Lambda}(x^*,\lambda^*)\hat{A}(x^*)^T & 0 & 0 \\ 0 & 0 & \overset{v}{D}(x^*,\lambda^*) \end{bmatrix} \quad .$$

In carrying out Newton's method for solving the nonlinear system $F(x,\lambda) = 0$, it is computationally convenient to retain the zero blocks in $F'(x^*,\lambda^*)$ at all points, not just at (x^*,λ^*). The resulting Newton-like method is still locally convergent with second-order asymptotic convergence rate. Given a starting point (x,λ) the linear system to be solved for the "Newton" correction is therefore

$$\begin{bmatrix} B & \hat{A} & \overset{\vee}{A} \\ \hat{\Lambda}\hat{A}^T & 0 & 0 \\ 0 & 0 & \overset{\vee}{D} \end{bmatrix} \begin{bmatrix} s \\ \hat{\delta} \\ \overset{\vee}{\delta} \end{bmatrix} = - \begin{bmatrix} g + \hat{A}\hat{\lambda} + \overset{\vee\vee}{A\lambda} \\ \hat{d} \\ \overset{\vee}{d} \end{bmatrix} . \qquad (3.14)$$

Before proceeding with the solution of system (3.14) it should be pointed out that generally it is not known which constraints will be active at a solution; therefore, the appropriate partitioning of the Jacobian in (3.14) is unknown. This is where a kind of active set strategy enters. At a given point x the constraints $c(x)$ are evaluated. Those constraints which are violated $(c_i(x) > 0)$ or exactly satisfied $(c_i(x) = 0)$ are considered active, the remaining constraints inactive. In finite precision arithmetic, a tolerance must be allowed for "exact" satisfaction of a constraint. Thus, for example, a constraint c_i may be deemed active at a point x if $c_i(x) \geq -\epsilon^{\frac{1}{2}}$, where ϵ is machine precision. It must be emphasized that this active constraint selection is used only in forming the linear system for computing the Newton step and plays no further role in subsequent line searches, where all constraints are taken into account (as will be seen shortly). It is this fact which helps avoid the combinatorial complexity and potential for jamming of many active set strategies.

An estimate for the Lagrange multiplier vector $\hat{\lambda}$ corresponding to the constraints considered active at x may be computed from system (3.11), just as for problem ECON. For the inactive constraints, of course, one would take $\overset{\vee}{\lambda} = 0$. Since $\lambda_i = 0$ and $c_i(x) < 0$ together imply that $d_i = 0$, it is seen from system (3.14) that $\overset{\vee}{\delta} = 0$. Thus, inactive constraints may as well be dropped from system (3.14), leaving the system

$$\begin{bmatrix} B & \hat{A} \\ \hat{\Lambda}\hat{A}^T & 0 \end{bmatrix} \begin{bmatrix} s \\ \delta \end{bmatrix} = - \begin{bmatrix} g + \hat{A}\hat{\lambda} \\ \hat{d} \end{bmatrix} . \tag{3.15}$$

The matrix of system (3.15) is not symmetric and the various numerical techniques used for system (3.2) are not directly applicable. However, assuming $\hat{\Lambda}$ is nonsingular, multiplying the lower row of both sides of equation (3.15) by $\hat{\Lambda}^{-1}$ yields the equivalent system

$$\begin{bmatrix} B & \hat{A} \\ \hat{A}^T & 0 \end{bmatrix} \begin{bmatrix} s \\ \delta \end{bmatrix} = - \begin{bmatrix} g + \hat{A}\hat{\lambda} \\ \hat{\Lambda}^{-1}\hat{d} \end{bmatrix} , \tag{3.16}$$

which is of the same form as (3.2). Since $\hat{c}(x) \geq 0$, the nonsingularity of $\hat{\Lambda}$ would follow if $\hat{\lambda} > 0$. At a solution the latter condition is implied by strict complementarity. Away from a solution, however, the estimate for $\hat{\lambda}$ given by (3.11) may have negative or zero components, which in turn could cause $\hat{\Lambda}$ to have zero or very small diagonal elements.

The geometric interpretation of a negative Lagrange multiplier λ_i in (3.11) is that the projection of the negative gradient $-g$ into the range space of A has a negative component in the direction of the gradient vector of the corresponding constraint c_i. This means that in the range space of A, f and c_i tend to decrease together, and therefore the i^{th} constraint places no restriction on movement from the current point which decreases f. Similarly, a zero Lagrange multiplier means, loosely speaking, that attainment of feasibility with respect to the corresponding constraint is unaffected by local movement along the projection of $-g$ into the range space of A. Thus, all constraints having negative or zero Lagrange multipliers may be dropped from the current active

set and the corresponding linear system (3.15), thereby assuring the nonsingularity of $\hat{\Lambda}$ and the validity of system (3.16).

Actually, it is not essential that constraints having zero Lagrange multipliers be deleted from the active set, which is perhaps comforting in that the motivation for doing so seems less clear than in the case of negative Lagrange multipliers. First, if $\lambda_i = 0$ but $c_i(x) > 0$ (the most likely case unless x is near a solution, where this difficulty should not arise), then the corresponding diagonal element of $\hat{\Lambda}$ is still positive. Even in the case where both λ_i and $c_i(x)$ are zero, however, the corresponding component of the right-hand side of (3.16) can still be defined by continuity, since $\lambda_i = 0$ and $c_i(x) \geq 0$ imply $\dfrac{d_i(x,\lambda)}{\{\Lambda(x,\lambda)\}_{ii}} = \dfrac{c_i(x)^2 + c_i(x)|c_i(x)|}{2(c_i(x) + |c_i(x)|)} = \dfrac{2c_i(x)^2}{4c_i(x)} = \dfrac{c_i(x)}{2} \to 0$ as $c_i(x) \to 0$. In practice, zero or near-zero Lagrange multipliers do not occur often for well-behaved problems, and it seems to make little difference in practice in the present algorithm whether the corresponding constraints are deleted or retained. A discussion of the effects of near-zero Lagrange multipliers is given by Gill and Murray (1974a), pp. 53-55.

It might be further argued along the same line that a constraint having a negative Lagrange multiplier need not be deleted unless it would otherwise lead to excessive cancellation error in computing the corresponding diagonal element of $\hat{\Lambda}$ (in particular, if $\hat{\Lambda}$ would be singular). Experience has shown, however, that retention of constraints having negative Lagrange multipliers can hamper progress toward a solution. As an extreme example, if n constraints are satisfied exactly, but some have negative Lagrange multipliers (so that the point is not a solution),

then further progress is impossible unless at least one constraint is deleted.

Unfortunately, it is not known until after solving the least-squares problem (3.11) which constraints, if any, may have negative or zero Lagrange multipliers. If a constraint is then deleted from the active set, the orthogonal-triangular factorization of $\hat{A}$ must be revised accordingly before the factorization is used in any further steps of the algorithm. Following deletion of a column from $\hat{A}$, the factorization can be updated very efficiently, requiring far fewer arithmetic operations than the original factorization (see Gill, Golub, Murray, and Saunders (1974)).

Since deletion of a single constraint from the active set generally alters the computed Lagrange multiplier estimates for all of the other constraints, it is unwise to delete several constraints simultaneously. If several constraints have negative Lagrange multipliers, then a reasonable strategy is to drop first the constraint having the most negative Lagrange multiplier, update the orthogonal factorization of $\hat{A}$, recompute the remaining Lagrange multipliers, then repeat the process if any of the new multipliers are negative.

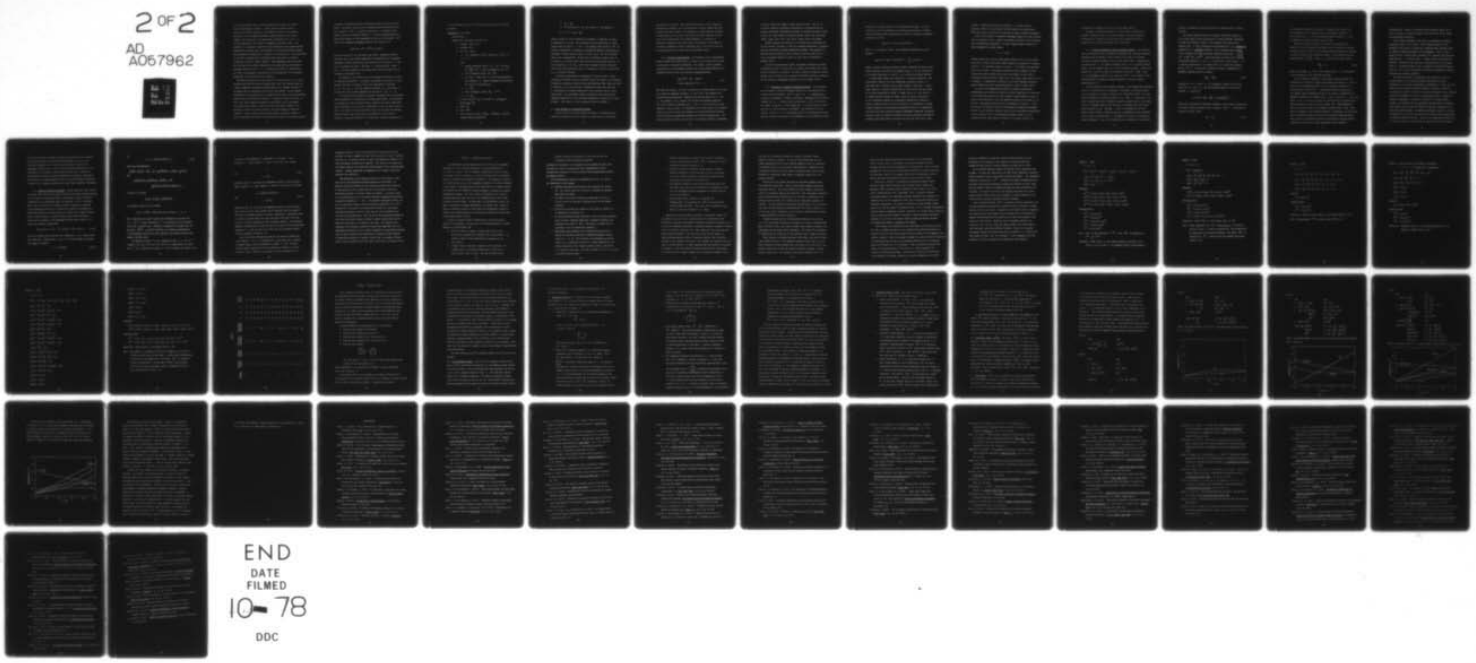
In order to apply the methods of Section 3.1.1 in solving system (3.16) it is required that $\hat{m} \leq n$. This is certainly a reasonable restriction for problem ECON, but for problem ICON it is entirely possible that more than n constraints may be violated at a given point and therefore considered active. In problem ECON, if there were exactly n constraints, the problem would become that of simply solving the nonlinear system $c(x) = 0$, which is precisely how Algorithm 3.1 would treat it. Analogously, in problem ICON, if there are more than n constraints violated at a given

point, then the problem is treated like that of solving the overdetermined system $\hat{c}(x) \cong 0$. At a solution to a nonlinear least-squares problem one would usually expect roughly half of the residuals to be positive and half negative. In practice the original nonlinear least-squares problem is not actually solved. Rather, after each iteration of a Gauss-Newton algorithm (e.g., Algorithm 2.5), any constraints which have ceased to be violated at the new point are deleted before carrying out the next iteration. In this way a number of constraints are dropped from the active set on each iteration until fewer than n constraints are active, at which point the usual algorithm for ICON is applicable. Usually, only a few iterations are required to reach this point. The Gauss-Newton step for $\hat{c}(x) \cong 0$ is given by the solution to the linear least-squares problem $\hat{A}^T s \cong -\hat{c}$, so that the orthogonal-triangular factorization of $\hat{A}^T$ is needed (rather than that of $\hat{A}$ as in Section 3.1.1). Thus, the general rule is that on every iteration the orthogonal factorization is computed for whichever of $\hat{A}$ and $\hat{A}^T$ has the larger number of rows. If $\hat{A}$ is square (i.e., if there are exactly n active constraints), then either branch of the algorithm should give the Newton step for the nonlinear system $\hat{c}(x) = 0$.

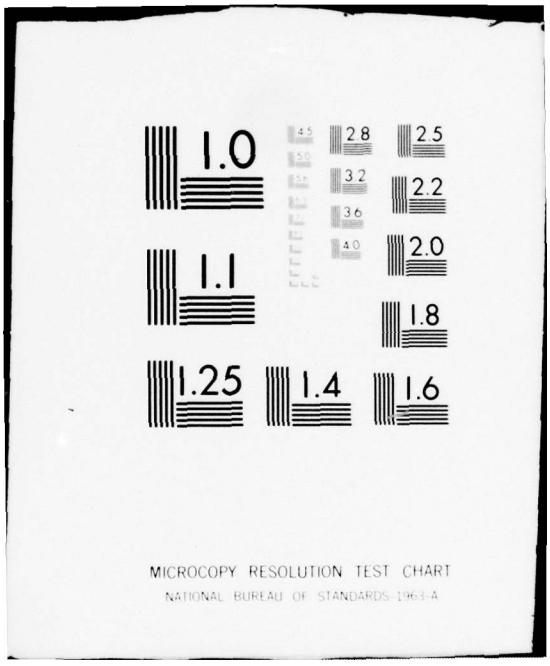
Having shown how to handle the alternative cases, we may now assume that $\hat{m} \leq n$ and $\lambda \geq 0$. The system (3.16) is solved as in (3.8) and (3.9), except that c is replaced by $\hat{\Lambda}^{-1}\hat{d}$, yielding Newton steps s_R and s_N for line searches in the range and null spaces of $\hat{A}$. If needed for a hybrid strategy, the corresponding gradients are $\hat{A}\hat{\Lambda}\hat{d}$ and $g + \hat{A}\hat{\lambda}$, respectively. It is important to note that the merit function for the range-space line search is $\varphi(x,\lambda) = \frac{1}{2}\|d(x,\lambda)\|^2$, not $\frac{1}{2}\|\hat{d}(x,\lambda)\|^2$, so

AD-A057 962 STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE F/6 12/1
NUMERICAL ALGORITHMS FOR NONLINEARLY CONSTRAINED OPTIMIZATION.(U)
APR 78 M T HEATH DAHC04-76-G-0185
UNCLASSIFIED STAN-CS-78-656 NL

2 OF 2
AD
A057962



END
DATE
FILMED
10-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

that all constraints--both active and inactive--are taken into account at all times during the search. A given inactive constraint has no effect on this merit function in the immediate vicinity of the current base point; if the constraint becomes violated during the line search, however, its contribution is automatically incorporated into $\varphi(x,\lambda)$ in a continuous manner, even though its Lagrange multiplier is zero. This behavior is in contrast to some active set strategies, which require special measures for detecting when a new constraint is encountered during a search and procedures for adding such a constraint to the active set. Also eliminated is the need to remember which constraints have been active on previous iterations in order to prevent jamming. These advantages stem mainly from the fact that in the present algorithm we are simply solving the system of nonlinear equations $F(x,\lambda) = 0$, a system which contains no inequalities, yet forces the inequality constraints and the nonnegativity of the Lagrange multipliers to be satisfied at its solution.

The merit function for the line search in the null space of $\hat{A}$ is also somewhat different from that for problem ECON. Since the feasible region in a purely inequality constrained problem usually has a nonempty interior, it can happen that no constraints are active at some points. In the standard formulation of the algorithm such a situation would result in the unconstrained minimization of the objective function f during the current iteration. In this event, since complete disregard for constraints in the line search could cause the new point to fall far outside the feasible region, it is necessary to add to the merit function a term which penalizes constraint violation. Such a penalty term is usually helpful in general, whether or not any constraints are active at the current base point, in avoiding excessive

violation of previously inactive constraints during the null-space line search. An obvious choice for such a penalized merit function would be $f(x) + \rho\varphi(x,\lambda)$, with ρ a positive scalar and φ as defined above. However, since the constraint function c is then evaluated anyway, for little more expense the full Lagrangian function can be evaluated. This leads to the augmented Lagrangian function (cf. Section 3.4.2)

$$L_A(x,\lambda,\rho) = f(x) + \lambda^T c(x) + \rho\varphi(x,\lambda)$$

as merit function for the null-space line search. Empirical evidence indicates that in the present algorithm for inequalities, this augmented Lagrangian is more effective and less sensitive to the choice of ρ than use of a penalty term alone. The value $\rho = \min\{1, |L/(2\varphi)|\}$, where L and φ are both evaluated at the current base point, has worked well in virtually all problems tried.

The methods of this section have been developed exclusively in terms of inequality constraints. If equality constraints are present in a given problem, they may be included in this same formulation by defining $d_i = c_i$ and $\Lambda_{ii} = 1$ for each equality constraint c_i, and taking c_i to be active at all points regardless of the sign of $c_i(x)$ or λ_i.

Our discussion of inequality constraints culminates with the following formal statement of an algorithm for solving problem ICON. For simplicity of presentation, only Newton-type search directions are shown explicitly; hybrid techniques, such as those of Algorithms 2.5 and 2.6, may be used as well. Also omitted from this algorithm schema are the tests and branches needed to bypass steps which may be unnecessary in a given context. Though not explicitly mentioned in the algorithm, equality constraints can

also be handled by means of the simple device given in the previous paragraph.

Algorithm 3.2 (for ICON)

Initial data: x

Repeat until convergence tolerance met

1. Evaluate $g(x)$, $c(x)$
2. Evaluate $\hat{A}(x)$
3. If $\hat{m} \geq n$ then
   1. s_R := minimum (or basic) solution to $\hat{A}^T s_R \cong -\hat{c}$
   2. $\lambda := 0$

   Else
   1. Compute orthogonal matrix $Q^T = (Y,Z)$ such that $Q\hat{A} = \begin{pmatrix} R \\ 0 \end{pmatrix}$ with R upper triangular as in (1.1)
   2. Solve triangular system $R\hat{\lambda} = -Y^T g$
   3. If $\hat{\lambda} \not\geq 0$, delete from $\hat{A}$ column corresponding to most negative λ_i, update orthogonal factorization, and return to preceding step
   4. $w := ZZ^T g$
   5. Solve triangular system $R^T p_R = -\hat{\Lambda}^{-1}\hat{d}$
   6. $s_R := Yp_R$
4. Do line search on α_R for descent on $\varphi(x+\alpha_R s_R, \lambda) = \frac{1}{2}\|d(x+\alpha_R s_R, \lambda)\|^2$
5. Form Bs_R
6. Form $Z^T BZ$
7. Solve symmetric system $Z^T BZ p_N = -Z^T(g+Bs_R)$ by block-diagonal-pivoting algorithm

8. $s_N := Zp_N$

9. Do line search on α_N for descent on $L_A(x+\alpha_N s_N, \lambda, \rho)$

10. $x := x + \alpha_R s_R + \alpha_N s_N$

Remarks similar to those following the statement of Algorithm 3.1 apply to Algorithm 3.2 as well. The convergence test for Algorithm 3.2 would require that the norms of w and d be suitably small and that Z^TBZ be positive semidefinite. Although an orthogonal-triangular factorization is performed in either branch of Step 3, in the $\hat{m} \geq n$ case the orthogonal matrix which triangularizes $\hat{A}^T$ need not be explicitly formed, since it is needed for no purpose other than solving the least-squares problem. As with Algorithm 3.1, the Bs_R term would be dropped from the right-hand side of the linear system in Step 7, if necessary, in order to ensure that s_N is a descent direction.

A careful implementation of Algorithm 3.2 should be able to solve optimization problems having an arbitrary mixture of equality and inequality constraints, including as special cases problems NLEQ, UCON, ECON, and ICON. In addition, due to the necessity of handling the case where more than n constraints are violated, Algorithm 3.2 also should be able to solve problem NLLS. An experimental computer program has been written which implements Algorithm 3.2 with sufficient generality to handle all of these problems. Test results for this program are given in Chapter 4.

3.4 Other Methods for Constrained Problems

In this section we review a number of methods for nonlinearly constrained optimization which are related in various ways to the methods

developed in this thesis. This survey is not meant to be an exhaustive compilation of methods, nor is the discussion of those methods which are included more than a sketch. The purpose is to acknowledge the influence of other research in the field and to place the ideas expressed in this thesis in the perspective of existing algorithms. In addition to individual references cited below, many of the methods are described in the books of Himmelblau (1972), Luenberger (1973), and Avriel (1976) and in survey papers such as Dixon (1975) and Fletcher (1977).

3.4.1 Successive Linearization. An alternative way of implementing Newton's method for problem ECON is to think of it as forming and solving a succession of linearized subproblems. For example, using a quadratic approximation to the Lagrangian function and a linear approximation to the constraint function gives the quadratic programming problem

$$\min_{s} \frac{1}{2} s^T B s + s^T w \quad \text{subject to the constraint} \quad A^T s + c = 0 \,. \tag{3.17}$$

Note that the solution to (3.17) is given by the linear system (3.2), which also yields the Lagrange multiplier δ for (3.17). This suggests the following general algorithm. Beginning with initial guesses for x and λ, the problem (3.17) is solved, its solution and Lagrange multiplier are used to update the estimates for x and λ, and the process is then repeated at the new point. If inequalities are present these can be handled either by use of an active set strategy or by incorporating them directly into the quadratic programming subproblem as linearized inequalities. The latter approach introduces the need for a full-fledged quadratic programming

subroutine rather than simply a linear equation solver. The use of successive quadratic programming subproblems for solving general nonlinearly constrained optimization problems was initiated by Wilson (1963) and has since been further developed by Murray (1969), Gill and Murray (1974a), Biggs (1972, 1975), Han (1976), Powell (1976), and others.

Another possible linearization is to linearize the constraints but not the objective function, so that the resulting subproblem is a general linearly constrained nonlinear minimization problem. This approach is taken by Rosen and Kreuser (1972), Robinson (1972) and Rosen (1977). A local convergence analysis is given for this class of algorithms by Robinson (1974).

As with all Newton-based methods, algorithms resulting from simple linearization cannot be relied upon to converge when started from a poor initial estimate for a solution. In order to improve robustness, special measures must be taken, such as penalty and barrier trajectory techniques for the quadratic programming subproblem case (see Wright (1976)).

3.4.2 Multiplier or Augmented Lagrangian Methods. An alternative way to attempt to solve the nonlinear system (3.1) for problem ECON is to solve the two equations $w(x,\lambda) = 0$ and $c(x) = 0$ separately and reconcile the different solutions iteratively. Beginning with an initial estimate for λ, a solution--which we may write as $x(\lambda)$--is found for the system $w(x(\lambda),\lambda) = 0$. The estimate for λ is then revised so that $c(x(\lambda)) \approx 0$, and the process is repeated. The solution to the subproblem $w(x(\lambda),\lambda) = 0$ is a stationary point of the Lagrangian function; however, because the Hessian of the Lagrangian is not necessarily positive

definite, the solution may not be an unconstrained minimum. In order to allow the use of techniques for unconstrained minimization, Hestenes (1969) and Powell (1969) independently introduced an augmented Lagrangian function of the form

$$L_A(x,\lambda,\rho) = L(x,\lambda) + \frac{1}{2}\rho c(x)^T c(x) \quad ,$$

where ρ is a positive scalar. This augmented Lagrangian has as its Hessian

$$W_A(x,\lambda,\rho) = W(x,\lambda) + \rho A(x)A(x)^T + \rho \sum_{i=1}^{m} c_i(x)G_i(x) \quad .$$

Using a classical theorem of Finsler (further elaborated by Dines, Afriat, Debreu, Herstein, and others--see Bellman (1970), p. 89, for references), it is easily shown that if (x^*,λ^*) is a solution for problem ECON and ρ is sufficiently large, then $W_A(x^*,\lambda^*,\rho)$ is positive definite. This result makes critical use of the optimality conditions, however, so that away from a solution to the overall problem, a value for ρ which makes $W_A(x,\lambda,\rho)$ positive definite may not exist, and in any case there is no direct indication how large ρ might need to be. The choice of ρ is rather delicate because too large a value can cause the Hessian to be ill conditioned (making the unconstrained minimization difficult), while too small a value can cause the Hessian to be indefinite or nearly singular (making the unconstrained minimization difficult or impossible). For this reason most practical augmented Lagrangian algorithms incorporate heuristic rules for adaptively refining the penalty parameter according to progress being made toward a solution (see Pierre and Lowe (1975), Glad

(1976)). Despite the difficulty of picking ρ, at least in this approach--unlike the usual penalty function methods--the penalty parameter need not go to infinity at a solution in order to assure convergence.

Much of the research on augmented Lagrangian methods has been concerned with the question of how to update the Lagrange multiplier vector following each unconstrained minimization. In their original papers Hestenes and Powell suggested the update formula

$$\lambda := \lambda + \rho c(x) \quad .$$

Several authors have used the least-squares system (3.11) in this context. With either of these first-order update formulas the asymptotic convergence rate for λ is generally linear, so long as ρ is bounded (see Bertsekas (1976) for a survey). Superlinear convergence may be obtained by using second-order update formulas which have been proposed by Buys (1972), O'Doherty and Pierson (1974), Tapia (1977), and others, and of which (3.4) is an example. A comprehensive discussion of all these Lagrange multiplier update formulas and their interrelationships is given by Tapia (1977).

In view of the computational expense of repeated unconstrained minimizations, an attractive variant of the above algorithm is to carry out the unconstrained minimizations only through a limited number of steps or until a loose tolerance is met (see Haarhoff and Buys (1970), Miele, et al (1972)). The logical extreme of such an approach is to update λ after each step of an iterative method for minimizing the augmented Lagrangian. In this case, if Newton's method is used for the unconstrained minimization step and formula (3.4) for the Lagrange multiplier update, then the algorithm becomes simply Newton's method for the larger system (3.1),

implemented via formulas (3.3) and (3.4) (see Tapia (1977)).

Although the multiplier method in its basic form is applicable to equality constraints only, it has been extended to include inequality constraints in a variety of ways by Buys (1972), Rockafellar (1973), Fletcher (1975), Glad (1976), Tapia (1977) and others.

3.4.3 Projected Gradient and Reduced Gradient Methods. These methods will be discussed in terms of equality constraints only, although inequality constraints may be handled as well by means of an active set strategy. The projected gradient method is due to Rosen (1960, 1961). Beginning with a feasible point x, the negative gradient vector $-g$ is projected orthogonally onto the tangent plane M_x to the constraint manifold. Recall from Sections 1.5.1 and 3.1.1 that this projection is given by $-ZZ^Tg$, where Z is the orthogonal basis for M_x given by (1.1a). A line search is then conducted along the negative projected gradient for descent (usually minimization) on f.

If the constraints are nonlinear, movement in the tangent plane generally leads to a new point which is infeasible. Therefore, a correction process must be carried out in order to return to the constraint surface. The step toward feasibility is along the direction Au, where u is given by (3.5). Since a single such step is unlikely to produce a point which is sufficiently near the constraint surface, the correction is repeated until some convergence tolerance is met. Although c must be re-evaluated at each new point in this iterative correction process, considerable work per iteration may be saved by using the same A throughout (at the price of having less rapid convergence). Once a new feasible point is obtained, a new projected

gradient is computed and the whole process is repeated until a solution is found.

The reduced gradient method for linearly constrained problems is due to Wolfe (1963) and was generalized to include nonlinear constraints by Abadie and Carpentier (1969). This algorithm also begins with a feasible point x whose components are partitioned into $n-m$ independent variables and m dependent variables. Thus, assuming proper ordering of the variables and using the obvious notation we may write $x = \begin{pmatrix} x_D \\ x_I \end{pmatrix}$, with $x_D \in \mathbb{R}^m$ and $x_I \in \mathbb{R}^{n-m}$. These subscripts are also used to indicate similar partitioning of g and A. In order that a small change s in x maintain feasibility it is required that, to a first-order approximation, $A^T s = 0$. Writing this condition in terms of independent and dependent variables gives the equation

$$A_D^T s_D = -A_I^T s_I \quad . \tag{3.18}$$

Assuming the $m \times m$ matrix A_D is nonsingular, system (3.18) may be solved for s_D given s_I. The corresponding change in the objective function f is given by

$$\Delta f \approx g^T s = g_I^T s_I + g_D^T s_D = (g_I - A_I A_D^{-1} g_D)^T s_I \quad .$$

Having thus eliminated the dependent variables, a line search is made along the negative reduced gradient vector $-(g_I + A_I \lambda)$, where λ is obtained by solving the linear system

$$A_D \lambda = -g_D \quad . \tag{3.19}$$

In evaluating the objective function f during the line search, the reduced gradient vector supplies only the independent variables s_I of the trial step s, and the remaining dependent variables s_D must be obtained by solving the linear system (3.18).

Since the new point produced by the line search generally is infeasible in the nonlinearly constrained case, an iterative correction procedure similar to that for the projected gradient algorithm is applied in order to return to the constraint surface. In the present case the correction step is $\begin{pmatrix} u_D \\ 0 \end{pmatrix}$, where u_D is given by the linear system

$$A_D^T u_D = -c \quad . \tag{3.20}$$

Again, for economy, A_D may be held constant while c is re-evaluated on each iteration of the correction procedure.

In this standard formulation of the reduced gradient method, the elimination of variables by simple partitioning of the matrix A can result in an ill-conditioned matrix A_D and consequent difficulty in solving systems (3.18)-(3.20). A numerically superior approach is to use properly chosen orthonormal bases for the spaces of independent and dependent variables (see Sargent (1974)). Thus, if we write $s = Ys_D + Zs_I$, where Y and Z are as in (1.1a), and repeat the above derivation, the reduced gradient is easily seen to be simply $Z^T g$. Transformation of $Z^T g$ into the full space of the original variables yields $ZZ^T g$, which will be recognized as the projected gradient. Furthermore, the correction step now becomes identical with that of the projected gradient method. Thus, there is no difference between the projected and reduced gradient methods when the latter is stably implemented using orthogonal

transformations. Indeed, the standard reduced gradient method can be viewed as merely a projected gradient method which uses oblique rather than orthogonal projections.

Still another algorithm of this type is the gradient restoration algorithm of Miele, et al (1969). In the "gradient phase" of this algorithm a line search is carried out, beginning from a feasible point x, along the negative gradient of the Lagrangian, $-w(x,\lambda)$, with λ given by (3.11). We have seen in Section 3.1.2, however, that with this definition for λ it follows that $w = ZZ^T g$, which is the projected gradient. Starting from the new point resulting from the line search, there is a "restoration phase" which is simply the correction procedure given by (3.5), so that the gradient restoration and gradient projection algorithms are formally identical.

The separate pursuit of minimization and feasiblity in the two distinct phases of the projected gradient method bears a strong resemblance to the approach represented by Algorithm 3.1. Thus, the line search along the negative projected gradient is similar in intent to the null-space search of Section 3.1.3, while the purpose of the correction procedure is analogous to that of the range-space search. Indeed, the variant of Algorithm 3.1 discussed in Section 3.2--in which Steps 1 through 7 are repeated until an approximate feasible point is obtained--would behave very much like the iterative correction procedure of the projected gradient method. An important difference, however, is the use in Algorithm 3.1 of the Newton step in the null-space search rather than the negative gradient direction. There are also important differences in the order in which the various steps are carried out and in how often

(and at what points) the problem functions and derivatives are evaluated. In particular, unlike the projected gradient method it is not at all necessary in Algorithm 3.1 to begin each null-space search from an approximate feasible point. The net effect of these differences is that Algorithm 3.1, by reducing to Newton-like behavior near a solution, is superlinearly convergent (possibly quadratically convergent, depending on how derivatives are approximated), while the projected gradient algorithm--at least in its standard formulation--has only linear asymptotic convergence.

3.4.4 Method of Bard and Greenstadt. Another algorithm which features a two-part line search, but one quite different from those discussed earlier, is due to Bard and Greenstadt (1969). This method is based on explicit recognition of the saddlepoint nature of solutions (x^*,λ^*) to problem ECON. Since a solution is in general neither a minimum nor a maximum of the Lagrangian function, it is difficult to monitor the progress of a Newton-like method for solving system (3.1) because it is unknown whether $L(x,\lambda)$ should decrease or increase along a given search direction. A Taylor series expansion of the Lagrangian function about a point (x,λ) yields the quadratic approximation

$$L(x+s,\lambda+\delta) \approx L + w^T s + c^T \delta + \frac{1}{2} s^T B s + s^T A \delta \equiv \Phi(s,\delta) , \qquad (3.21)$$

where all functions and derivatives are evaluated at (x,λ) unless otherwise indicated. Assuming that A and B are of full rank, we now make the change of variables

$$\bar{s} = s + B^{-1}(A\delta+w) \qquad (3.22)$$

and

$$\bar{\delta} = \delta + (A^T B^{-1} A)^{-1}(A^T B^{-1} w - c) \quad . \tag{3.23}$$

Under this transformation

$$\frac{1}{2}\bar{s}^T B \bar{s} = \frac{1}{2} s^T B s + s^T A \delta + w^T s + \frac{1}{2}\delta^T A^T B^{-1} A \delta + w^T B^{-1} A \delta + \frac{1}{2} w^T B^{-1} w$$

and

$$\frac{1}{2}\bar{\delta}^T A^T B^{-1} A \bar{\delta} = \frac{1}{2}\delta^T A^T B^{-1} A \delta + \delta^T A^T B^{-1} w - \delta^T c$$

$$+ \frac{1}{2}(A^T B^{-1} w - c)^T (A^T B^{-1} A)^{-1} (A^T B^{-1} w - c) \quad ,$$

so that, if we define

$$\bar{\Phi}(\bar{s}, \bar{\delta}) = \frac{1}{2}\bar{s}^T B \bar{s} - \frac{1}{2}\bar{\delta}^T A^T B^{-1} A \bar{\delta} \quad ,$$

the quadratic function (3.21) becomes

$$\Phi(s, \delta) = \bar{\Phi}(\bar{s}, \bar{\delta}) + \text{terms which do not involve } s \text{ or } \delta \, .$$

This "completing the square" process has eliminated the bilinear term in s and δ, thereby expressing Φ as a difference of pure quadratic forms plus a constant term. Therefore, assuming for the moment that B, and hence $A^T B^{-1} A$, is positive definite, the quadratic approximation to the Lagrangian function has been separated into positive definite and negative definite parts.

The quadratic function $\bar{\Phi}$ has a saddlepoint when $\bar{s} = 0$ and $\bar{\delta} = 0$ (compare (3.22) and (3.23) with (3.3) and (3.4), respectively), at which point $\bar{\Phi}$ has a minimum with respect to $\bar{s}$ and a maximum with respect to $\bar{\delta}$.

The step to this saddlepoint is separated into two parts. First, setting $\bar{s} = 0$ while holding $\bar{\delta}$ fixed in (3.22) and (3.23) yields

$$\delta = 0$$

and (3.24)

$$s = -B^{-1}w \quad .$$

Along this step Φ, and hence the Lagrangian function, should decrease. Next, setting $\bar{\delta} = 0$ while holding $\bar{s}$ fixed in (3.22) and (3.23) yields

$$\delta = -(A^TB^{-1}A)^{-1}(A^TB^{-1}w-c)$$

and (3.25)

$$s = -B^{-1}A\delta \quad .$$

Along this step Φ, and hence the Lagrangian function, should increase. Thus, each of the two line searches can be conducted with the clear-cut objective of attaining descent or ascent, respectively. Sufficiently near a solution to problem ECON, full steps are taken along both (3.24) and (3.25), so that the algorithm reverts to Newton's method. An interesting geometric interpretation is that the step (3.24) moves to the unconstrained minimum of Φ, then the step (3.25) moves onto the constraint surface (with an attendant increase in Φ).

All of the above developments depend on the positive definiteness of the matrix B. If B is indefinite at a point away from a solution to problem ECON, it may be reasonable to replace B with a positive definite matrix obtained by one of the methods in Section 2.4.1, then proceed as usual. However, for problems in which the Hessian of the

Lagrangian function is not positive definite at a solution--not only possible, but quite commonly the case--such a practice is open to question and would not, in general, result in Newton-like asymptotic behavior, nor would convergence necessarily be expected. This could be one reason why this method, despite its theoretical attractiveness, has not attained great currency. Perhaps augmenting the Lagrangian with a penalty term would alleviate this difficulty.

The similarities of the methods discussed in this section to each other and to the algorithms developed earlier are no accident. As Fletcher (1977) has pointed out, the borrowing of ideas which work well in one method for use in another has led to a blurring of the usual distinctions between the various classes of methods. A consensus seems to be developing--the use of successive quadratic programming subproblems, simultaneous updating of x and λ in augmented Lagrangian methods, the use of second-order terms in projected gradient algorithms, etc.--which indicates an ever greater use of Newton-like methods for constrained optimization problems, with the result that all these algorithms take on a more uniform appearance. For this reason the algorithms developed in this thesis are based directly on Newton's method, thereby allowing the problem of robustness to be tackled in the open. This approach also simplifies the convergence analysis of Newton-like algorithms which use approximate derivatives, such as finite differences or quasi-Newton methods, since it makes possible direct application of Kantorovich-type analyses for Newton-like methods in a general context (see, for example, Robinson (1974), Dennis and Moré (1977), Han (1976), Glad (1976), and Tapia (1977)).

CHAPTER 4. COMPUTATIONAL RESULTS

An experimental computer program has been written which implements Algorithm 3.2 with sufficient generality to subsume Algorithms 2.5, 2.6, and 3.1 as well. Thus the program is applicable to all five problem classes listed in Section 1.3 and also to problems with an arbitrary mixture of equality and inequality constraints. This program is intended as a pilot project to test the fundamental validity of the ideas developed in this thesis. The program has not had the benefit of a lengthy development process nor the exhaustive testing necessary to produce high quality mathematical software. For these reasons the program is in no sense a polished production code, nor will it be documented here or the coding discussed in great detail. In particular, no extraordinary efforts were made to economize on storage requirements or execution time. Indeed, a number of potential savings in these areas were deliberately forgone in order to avoid complications in the coding which might have obscured some of the larger points at issue.

It was not possible within a limited time to fully develop all aspects of the program. In particular, the following features are implemented in a rather minimal way:

1. The trust radius is simply a liberal upper bound on step size, fixed throughout all iterations for a given problem. No attempt is made to refine adaptively the estimate for the trust radius.
2. A simple descent strategy is employed in line searches. The original step is successively halved until a point with a lower function value is found. The more exacting descent

criteria discussed in Section 2.4.4 are not used, nor are interpolatory point-generating algorithms.

Although the performance of the program on test problems has been quite satisfactory, there is evidence that full implementation of more sophisticated strategies in these two phases would significantly enhance performance further.

The following major features of Algorithms 2.5, 2.6, 3.1, and 3.2 are implemented in the program:

1. Separate range-space and null-space line searches are carried out. (Of course, only one of the two is relevant for problems NLEQ, NLLS, and UCON.)
2. The hybrid curve-search technique of Algorithms 2.5 and 2.6 is used in all line searches (including those for constrained problems).
3. Inequality constraints are handled by the method of Theorem 3.2, as implemented in Algorithm 3.2.
4. The block-diagonal-pivoting algorithm is used for solving symmetric linear systems, computing directions of negative curvature, and computing the inertia of Z^TBZ. (The latter is needed for the convergence test for minimization problmes.)
5. Householder transformations with column pivoting are used to compute the (possibly truncated) orthogonal-triangular factorization of A or A^T. The criterion for truncating the factorization (i.e., estimating the rank) is a simple threshold for the absolute magnitude of the largest unreduced column. The matrix Z is obtained explicitly. The basic solution to $A^Ts \cong -c$ is used for problems NLEQ and NLLS.

6. Analytic expressions are required for all first derivatives, but the second-derivative matrix Z^TBZ is obtained by finite differences along the columns of Z.

7. Violated inequality constraints having negative estimated Lagrange multipliers are deleted from the active constraint set, while violated inequality constraints having near-zero multipliers are retained in the active set.

8. The objective function is used as the merit function for the null-space line search if all constraints are equalities, otherwise the Lagrangian function (augmented by a penalty term) is used in this capacity.

9. The convergence test consists of a tolerance for $\max\{\|w\|_\infty, \|d\|_\infty\}$ and requiring that Z^TBZ be positive semi-definite (where relevant). This works for all problems except NLLS, for which the tolerance is on $\|Ac\|_\infty$.

Empirical testing of numerical software is notoriously complex for any problem area, and this is especially true for optimization codes. Differences in machine enviornments--arithmetic precision, compilers, etc.--as well as fundamental differences in algorithm design--convergence criteria, order of derivatives required, etc.--make comparison with previously published results difficult. Even with all of these factors equal, the highly problem-dependent nature of algorithm behavior often frustrates attempts to draw definite conclusions. For example, the relative effectiveness of various options in a single algorithm may vary greatly from problem to problem, or even with different starting points for the same problem. Such effects may be averaged somewhat over a statistical ensemble, but in

this case it is difficult to define or select an unbiased, random sample of realistic problems. In view of these difficulties and the rather preliminary nature of the computer program at hand, an exhaustive and detailed account of test results seems pointless. Therefore, only a qualitative discussion of the overall testing procedure will be given, followed by a relatively small number of numerical results for a few typical problems.

Most of the test problems selected are well-known examples taken from published collections. Some are lesser-known examples taken from individual sources. Only those problems for which analytic derivatives could be conveniently supplied were selected. This requirement eliminated mainly problems having discontinuities in functions or derivatives and problems whose functions are defined only by complicated computer subroutines. For each problem all published starting points were used and in most cases several additional starting points were chosen, usually so as to have some form of difficulty, such as having a rank-deficient Jacobian or being relatively far from a solution.

Although the main goal of the algorithms developed in this thesis is to solve constrained problems, the program was tested on problems NLEQ, NLLS, and UCON as well, since these problems allow separate testing of the range-space and null-space search strategies. The results also confirmed the feasibility of solving all of these problems with a single subroutine.

Forty systems of nonlinear equations were solved, about half of these being systems of order 2, the remainder ranging in order from 3 up to 10. These problems were taken largely from collections such as Bus (1975) and Hardaway (1968) and included the standard well-known examples of Brown, Broyden, Powell, etc. The nonlinear least-squares problems were 27 in

number and were taken mainly from the collections of Gill and Murray (1976) and Nash (1976). The largest such problem considered involved fitting a function with five parameters to 33 data points. All of the NLLS problems were also solved by unconstrained minimization (thereby exercising the null-space branch of the program rather than the range-space branch). Twelve additional unconstrained problems were also used in testing.

There were a total of 42 constrained optimization problems tested. These were taken from collections such as Asaadi (1973), Bracken and McCormick (1968), Colville (1968), Himmelblau (1972), Miele, et al (1971, 1972), Pierre and Lowe (1975), and Wright (1976), as well as numerous individual sources. The constrained problems included 14 with equality constraints only, 16 with inequalities only, and 12 having both equality and inequality constraints. The number of variables in these problems ranged from 3 to 20 and the number of constraints from 1 to 40. Although the algorithm employed takes no advantage of linear constraints (or of more special structure such as upper and lower bounds on variables), several linearly constrained problems (11 of the 42) were solved. These linearly constrained problems were especially helpful in debugging the program.

With so many test problems and with a half dozen or more starting points for each, a complete statement of the problems and a detailed discussion of results of all test runs would require an entire volume. A fair summary of the results, however, is that the program converged to a correct solution from all of the starting points for most of the problems and from most of the starting points for virtually all of the problems. Thus, the methods developed here appear to show great promise as the basis for a robust algorithm for a variety of optimization problems. Less effort was spent on the efficiency of the program in obtaining solutions, and precise comparisons with other

methods are difficult to make (for reasons outlined earlier), but the performance of the program in this regard was usually satisfactory, ranging from quite good on several problems to fairly poor on a few.

We now present numerical results for several of the constrained test problems. All test runs were made on the IBM 370/168 computers at Stanford Linear Accelerator Center using double precision arithmetic. The convergence tolerance in each case is 10^{-6}. In Table 2 the number of range-space and null-space line searches required to attain this accuracy is given, along with the number of times the problem functions and their derivatives were evaluated. The counts for the two line searches may differ for a given problem because either search may be skipped on a given iteration if the current point is already within the relevant tolerance.

Since the Hessian matrix Z^TBZ is obtained by finite differences, the necessary gradient evaluations are included in--and are usually a sizeable fraction of--the total count. Allowance for this must be made in comparing the present results with similar tests of methods which use analytically defined second derivatives and therefore do not need extra gradient evaluations for this purpose. Moreover, since the program evaluates the columns of $A(x)$ one at a time, the figure quoted is the total number of such individual constraint gradients evaluated. Thus, for an equality constrained problem, the figure given must be divided by m to obtain the total number of times $A(x)$ was evaluated. In the inequality case only gradients of active constraints are evaluated at each iteration.

Example 1: ECON

$n = 5,\ m = 3$

$$f(x) = (x_1-1)^2 + (x_1-x_2)^2 + (x_2-x_3)^3 + (x_3-x_4)^4 + (x_4-x_5)^4$$

$$c_1(x) = x_1^2 + x_2^2 + x_3^3 - 2 - 3\sqrt{2} = 0$$
$$c_2(x) = x_2 - x_3^2 + x_4 + 2 - 2\sqrt{2} = 0$$
$$c_3(x) = x_1 x_5 - 2 = 0$$

Solutions:

$$x^{*(1)} \approx (1.117,\ 1.220,\ 1.538,\ 1.973,\ 1.791)^T$$
$$x^{*(2)} \approx (-2.791,\ 3.004,\ 0.205,\ 3.875,\ -0.717)^T$$
$$x^{*(3)} \approx (-1.273,\ 2.410,\ 1.195,\ -0.154,\ -1.571)^T$$
$$x^{*(4)} \approx (-0.703,\ 2.636,\ -0.0964,\ -1.798,\ -2.843)^T$$

Starting points:

$$x^{(1)} = (1,1,1,1,1)^T$$
$$x^{(2)} = (2,2,2,2,2)^T$$
$$x^{(3)} = (-1,3,-0.5,-2,-3)^T$$
$$x^{(4)} = (-1,2,1,-2,-2)^T$$
$$x^{(5)} = (0,0,0,0,1)^T$$

Note: $A(x)$ is rank deficient at $x^{(5)}$, while Z^TBZ is indefinite at $x^{(4)}$ and $x^{(5)}$.

Reference: Wright (1976), p. 228 (problem modified from Miele, et al (1971), p. 127, so that f is unbounded below if unconstrained.)

Example 2: ECON

$n = 5,\ m = 3$

$$f(x) = x_1x_2x_3x_4x_5$$

$$c_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$
$$c_2(x) = x_2x_3 - 5x_4x_5 = 0$$
$$c_3(x) = x_1^3 + x_2^3 + 1 = 0$$

Solutions:

$$x^{*(1)} \approx (-1.717,\ 1.596,\ 1.827,\ -0.764,\ -0.764)^T$$
$$x^{*(2)} \approx (-0.699,\ -0.870,\ -2.790,\ -0.697,\ -0.697)^T$$

Starting points:

$$x^{(1)} = (-2,2,2,-1,-1)^T$$
$$x^{(2)} = (-1,-1,-1,-1,-1)^T$$
$$x^{(3)} = (-2,-2,-2,-2,-2)^T$$
$$x^{(4)} = (0.001,\ 1.0,\ 2.774,\ 1.0,\ 0.5548)^T$$

References: Powell (1969), p. 287; Wright (1976), p. 223.

Note: Powell originally used $f(x) = \exp(x_1x_2x_3x_4x_5)$ as objective function so that f would be bounded below. This modification is unnecessary in the present algorithm. The matrix Z^TBZ is indefinite at $x^{(4)}$, which was used by O'Doherty and Pierson (1974), p. 194.

Example 3: ICON

$n = 4,\ m = 3$

$$f(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4$$

$$c_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 - 8 \leq 0$$

$$c_2(x) = x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 - 10 \leq 0$$

$$c_3(x) = 2x_1^2 + x_2^3 + x_3^2 + 2x_1 - x_2 - x_4 - 5 \leq 0$$

Solution:

$$x^* = (0,1,2,-1)^T$$

Starting points:

$$x^{(1)} = (0,0,0,0)^T$$

$$x^{(2)} = (3,3,3,3)^T$$

References: Rosen and Suzuki (1965), p. 113; Wright (1976), p. 243.

Note: Starting point $x^{(2)}$ was used by Glad (1976), p. 67.

Example 4: Mixed equality and inequality constraints

n = 3, m = 5 (2 equalities, 3 inequalities)

$$f(x) = 1000 - x_1^2 - 2x_x^2 - x_3^2 - x_1x_2 - x_1x_3$$

$$c_1(x) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$

$$c_2(x) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

$$c_3(x) = -x_1 \leq 0$$

$$c_4(x) = -x_2 \leq 0$$

$$c_5(x) = -x_3 \leq 0$$

Solution:

$$x^* \approx (3.512,\ 0.217,\ 3.552)^T$$

Starting points:

$$x^{(1)} = (2,2,2)^T$$

$$x^{(2)} = (10,10,10)^T$$

$$x^{(3)} = (-5,-10,5)^T$$

References: Himmelblau (1972), p. 397 (problem attributed to D. A. Paviani); Wright (1976), p. 220.

Example 5: ICON

$n = 9,\ m = 14$

$$f(x) = -0.5(x_2x_6 - x_1x_7 + x_3x_7 + x_5x_8 - x_4x_9 - x_3x_8)$$

$$c_1(x) = x_1^2 + x_6^2 - 1 \leq 0$$

$$c_2(x) = (x_2-x_1)^2 + (x_7-x_6)^2 - 1 \leq 0$$

$$c_3(x) = (x_3-x_1)^2 + x_6^2 - 1 \leq 0$$

$$c_4(x) = (x_1-x_4)^2 + (x_6-x_8)^2 - 1 \leq 0$$

$$c_5(x) = (x_1-x_5)^2 + (x_6-x_9)^2 - 1 \leq 0$$

$$c_6(x) = x_2^2 + x_7^2 - 1 \leq 0$$

$$c_7(x) = (x_3-x_2)^2 + x_7^2 - 1 \leq 0$$

$$c_8(x) = (x_4-x_2)^2 + (x_8-x_7)^2 - 1 \leq 0$$

$$c_9(x) = (x_2-x_5)^2 + (x_7-x_9)^2 - 1 \leq 0$$

$$c_{10}(x) = x_3^2 - 1 \leq 0$$

$$c_{11}(x) = (x_4-x_3)^2 + x_8^2 - 1 \leq 0$$

$$c_{12}(x) = (x_5-x_3)^2 + x_9^2 - 1 \leq 0$$

$$c_{13}(x) = x_4^2 + x_8^2 - 1 \leq 0$$

$$c_{14}(x) = (x_4-x_5)^2 + (x_9-x_8)^2 - 1 \leq 0$$

$$c_{15}(x) = x_5^2 + x_9^2 - 1 \leq 0$$

$$c_{16}(x) = -x_1 \leq 0$$

$$c_{17}(x) = -x_6 \leq 0$$

$$c_{18}(x) = -x_5 \leq 0$$

$$c_{19}(x) = x_1 - x_2 \leq 0$$

$$c_{20}(x) = -x_7 \leq 0$$

$$c_{21}(x) = x_2 - x_3 \leq 0$$

$$c_{22}(x) = x_5 - x_4 \leq 0$$

$$c_{23}(x) = x_8 \leq 0$$

$$c_{24}(x) = x_9 \leq 0$$

$$c_{25}(x) = x_4 - x_3 \leq 0$$

Solutions:

$$x^{*(1)} \approx (0.402,\ 0.939,\ 1.0,\ 0.939,\ 0.402,\ 0.5,\ 0.344,\ -0.344,\ -0.5)^T$$

$$x^{*(2)} \approx (0.257,\ 0.580,\ 1.0,\ 0.794,\ 0.156,\ 0.236,\ 0.396,\ -0.608,\ -0.537)^T$$

Starting points:

$$x^{(1)} = (0.35,\ 1.0,\ 1.1,\ 1.0,\ 0.33,\ 0.6,\ 0.45,\ -0.47,\ -0.7)^T$$

$$x^{(2)} = (0.1,\ 0.125,\ 0.167,\ 0.143,\ 0.111,\ 0.2,\ 0.25,\ -0.2,\ -0.25)^T$$

References: Murray (1969), p. 85; Wright (1976), p. 245.

Note: The problem is to maximize the area of a hexagon no two of whose vertices are more than one unit apart. Several other configurations also have the same maximum area. The matrix Z^TBZ is indefinite at both starting points. An alternate version of this problem, which was also used in testing, is given by Himmelblau (1972), p. 415, and by Lootsma (1972), p. 375.

TABLE 2

Problem	Starting Point	Solution Found	Range-space Searches	Null-space Searches	f	g	c	Columns of A
1	1	1	8	6	15	32	16	87
	2	1	8	7	15	32	16	96
	3	4	8	7	18	32	16	96
	4	3	7	7	15	31	15	93
	5	1	16	15	35	65	32	191
2	1	1	5	4	9	20	10	60
	2	2	6	4	10	24	13	72
	3	2	7	5	12	28	14	84
	4	1	8	7	15	32	16	96
3	1	1	10	11	42	53	52	71
	2	1	18	18	60	82	78	137
4	1	1	7	5	13	19	23	40
	2	1	8	7	24	23	32	48
	3	1	13	10	40	35	55	74
5	1	1	9	8	22	44	31	269
	2	2	15	16	82	104	97	412

APPENDIX. OPERATION COUNTS

Since arithmetic operation counts for algorithms can make especially dull reading and need be absorbed in detail only by the most interested of readers, these are isolated in this appendix so as not to disrupt the continuity of the main body of this thesis. We are concerned specifically with the relative efficiencies of various methods for solving the linear system (3.2). As usual, only the dominant term of each operation count is given (in this case terms of degree three in n and m), and one unit is considered to be a multiplication or division coupled with an addition or subtraction.

Several strategies for solving system (3.2) are possible:

1. Solve the full system (3.2) directly.
2. Form and solve systems (3.4) and (3.3).
3. Form and solve systems (3.5), (3.6), and (3.7).
4. Form and solve systems (3.8), (3.9), and (3.10).
5. Form and solve systems

$$\begin{pmatrix} Z^T B \\ A^T \end{pmatrix} s = - \begin{pmatrix} Z^T w \\ c \end{pmatrix}$$

and (3.7), where Z is an $n \times (n-m)$ matrix whose columns form a basis for the null space of A.

(Note that Method 4 is a special case of Method 3 having a particular choice for the matrix Z.)

The principal factor which determines the relative efficiencies of these methods is the trade-off between the cost of forming the smaller systems and the cost of solving those systems. In particular, any matrix

multiplications or factorizations required in forming a linear system must be included in the operation count along with the cost of solving that system. This trade-off point is in turn greatly affected by the relative sizes of m and n. For this reason, comparisons between methods will be stated in terms of the ratio $\beta = m/n$, $0 \leq \beta \leq 1$. Although some consideration will be given to numerical stability, this appendix is primarily concerned with efficiency. The five methods listed above vary considerably in robustness and potential accuracy. In particular, if the matrix A is deficient in rank, then the matrix of system (3.2) is singular and all five methods would fail in principle. In this event, if the orthogonal factorization is implemented with the additional overhead necessary for column pivoting (see Section 1.5.1), Method 4 offers a means of recovery not readily available in the other methods. It seems unfair, however, to penalize Method 4 for its ability to cope with the rank-deficient case when the other methods are unable to do so reliably. Therefore, the five methods are compared on their common domain of applicability, full-rank problems.

We first consider some of the subtasks relevant to one or more of the methods.

A.1 Solving Linear Systems. Since all of the symmetric matrices which arise in this context may be indefinite, the corresponding linear systems are solved by means of the block-diagonal-pivoting algorithm (see Section 1.5.2). For a symmetric matrix of order k, this algorithm requires about $k^3/6$ arithmetic operations. The square, nonsymmetric systems which arise are solved by means of the LU factorization resulting from Gaussian elimination with partial pivoting (see Forsythe and Moler (1967)).

For a matrix of order k, this algorithm requires about $k^3/3$ arithmetic operations.

A.2 Forming the Matrix Z. In addition to the orthogonal-triangular factorization of A, several alternative techniques based on Gaussian elimination have been proposed for computing the basis for the null space of A represented by the matrix Z.

1. Orden (1964) suggested the use of Gauss-Jordan elimination to reduce A^T to the form

$$MA^T = (I_m, C) ,$$

where I_m is the $m \times m$ identity matrix and C is $m \times (n-m)$. Then let

$$Z = \begin{pmatrix} -C \\ I_{n-m} \end{pmatrix} .$$

This process requires about $nm^2 - m^3/2$ operations, or $(\beta^2 - \beta^3/2)n^3$.

2. Wolfe (1967) suggested appending $n-m$ additional linearly independent rows--comprising an $(n-m) \times m$ matrix C^T--to A^T, then letting Z be the last $n-m$ columns of $\begin{pmatrix} A^T \\ C^T \end{pmatrix}^{-1}$. (In Wolfe's scheme the columns of C are selected from among the gradients of inactive constraints, which are in adequate supply because of an assumed nonnegativity constraint on x.) The straightforward use of Gaussian elimination and back-substitution to compute this portion of the explicit inverse would require about $4n^3/3 - n^2m$ operations. However, by taking advantage of the special nature of the right-hand sides

(i.e, columns of the identity matrix; see Isaacson and Keller (1966), p. 36), the work can be reduced to $n^3 - n^2m + nm^2/2 - m^3/6$ operations, or $(1 - \beta - \beta^2/2 - \beta^3)n^3$.

3. McCormick (1970) suggested partitioning the columns of A^T (reordering them if necessary) so that $A^T = (C_1, C_2)$, with C_1 $m \times m$ and nonsingular. Then let

$$Z = \begin{pmatrix} -C_1^{-1}C_2 \\ I_{n-m} \end{pmatrix}.$$

This process requires about $nm^2 - 2m^3/3$ operations, or $(\beta^2 - 2\beta^3/3)n^3$. Note that this method is effectively the same as that of Orden (1964) and produces a matrix Z having the same special structure. By using a complete pivoting strategy in the Gauss-Jordan reduction, however, Orden's method provides an explicit mechanism for selecting m linearly independent columns of A^T, rather than simply assuming such a selection is known in advance.

4. The orthogonal-triangular factorization of A requires about $nm^2 - m^3/3$ operations. However, the matrix Z is not made explicitly available from the factorization algorithm alone, since the orthogonal matrix $Q = \begin{pmatrix} Y^T \\ Z^T \end{pmatrix}$ is implicitly represented by the sequence of Householder transformations which triangularize A. In a given context it may or may not be advantageous to leave Z in this implicit form (see the discussion of this issue in Section 3.1.1). If Z is desired in explicit form, the standard algorithm--alluded to on p. 13--for multiplying out the Householder

transformations requires about $2n^2m - 3nm^2 + m^3$ operations. Recently, Kaufman (1977) has devised an algorithm for this purpose--Algorithm B in her paper--which requires $n^2m - nm^2/2 - m^3/3$ operations. The latter is about the same cost as for the standard algorithm--which Kaufman calls Algorithm A--for $m \approx n$, but only about half the cost of the standard algorithm for $m \ll n$. Thus, the total cost of producing Z explicitly by orthogonal factorization is about $n^2m + nm^2/2 - 2m^3/3$ operations, or $(\beta + \beta^2/2 - 2\beta^3/3)n^3$.

For this particular subtask the question of numerical stability may be an overriding factor, since the three elimination methods have serious deficiencies in this regard. In particular, in the Gauss-Jordan reduction required by Orden's method, even with complete pivoting, the row multipliers may become large (see Isaacson and Keller (1966), pp. 50-51; this behavior is in contrast to ordinary Gaussian elimination with partial pivoting, where the multipliers are bounded by unity), with an attendant potential for error amplification. In the methods of Wolfe and McCormick it is difficult to choose the columns of C and C_1, respectively, so that the resulting matrices are nonsingular and well-conditioned. For this reason the orthogonal factorization of A is probably the method of choice in most cases for computing Z. For purposes of comparing efficiency, however, the possibility of using an elimination method for computing Z is included as an option in Methods 3 and 5 in the complete operation counts given below. The specific elimination method chosen is the Gauss-Jordan reduction method of Orden, since it seems to be the most stable and straightforward to implement and has almost the same efficiency as McCormick's method.

A.3 Forming the Matrix Z^TBZ. This issue is discussed in some detail in Section 3.1.1. There are three distinct cases.

1. Matrix multiplication. If both B and Z are explicitly available, then the product Z^TBZ may be formed by ordinary matrix multiplication. Taking advantage of the fact that Z^TBZ is symmetric, this process requires $3n^3/2 - 2n^2m + nm^2/2$ operations, or $(3/2 - 2\beta + \beta^2/2)n^3$, if Z is dense. For the Z resulting from either of the elimination methods of Orden (1964) and McCormick (1970), the special structure of Z can be exploited to reduce this cost to $3n^2m/2 - 2nm^2 + m^3/2$ operations, or $(3\beta/2 - 2\beta^2 + \beta^3/2)n^3$.

2. Congruence transformations. If B is explicitly available but Z is represented implicitly, then the sequence of Householder transformations may be applied to the symmetric matrix B as congruence transformations, producing Z^TBZ in an efficient manner (see Stewart (1973), p. 335). This process requires about $2n^2m - nm^2/2$ operations, or $(2\beta - \beta^2/2)n^3$, while saving the work that would be necessary to form Z explicitly.

3. Finite differences. If Z is explicitly available but B must be approximated by finite differences, then BZ may be approximated directly by finite differences along the columns of Z, thereby saving a matrix multiplication as well as m gradient evaluations. The product $Z^T(BZ)$ is then formed by a single matrix multiplication. Because of the approximate nature of BZ in this case, symmetry should not be assumed. Rather, the entire square array Z^TBZ should be formed, then symmetrized by

averaging with its transpose. Thus the total cost is $n^3 - 2n^2m + nm^2$ operations, or $(1 - 2\beta + \beta^2)n^3$, for dense Z. Again, for the special Z resulting from elimination the cost reduces to $n^2m - 2nm^2 + m^3$, or $(\beta - 2\beta^2 + \beta^3)n^3$. Of course, none of these figures include any work which may be involved in the function evaluations needed to form BZ.

An analogous discussion to the above applies to the formation of the matrix Z^TB needed in Method 5. The costs in this case are $n^3 - n^2m$ for matrix multiplication ($n^2m - nm^2$ for the special Z resulting from elimination), $2n^2m - nm^2/2$ for direct use of Householder transformations (there is no gain over the Z^TBZ case bacause of the lack of symmetry), and, of course, zero for finite differences along the columns of Z.

A.4 Forming the Matrix $A^TB^{-1}A$. The matrix $A^TB^{-1}A$ is needed only in Method 2. First, the block-diagonal factorization of B--which is also needed in solving system (3.3)--is computed by the block-diagonal-pivoting algorithm at a cost of $n^3/6$ operations. Then the m triangular systems having the correspondingly transformed columns of A as right-hand sides are solved by back-substitution in order to form $B^{-1}A$. This step requires about n^2m operations. Finally, the product $A^TB^{-1}A$ is formed by matrix multiplication, taking advantage of symmetry, at a cost of $nm^2/2$ operations. Thus, the total cost is about $n^3/6 + n^2m + nm^2/2$ operations, or $(1/6 + \beta + \beta^2/2)n^3$.

A.5 Total Costs. We are now in a position to give the full operation counts for Methods 1 through 5. Methods 1 and 2 can be carried out in only one way. A variety of options are available in implementing Methods 3, 4, and 5, however, making comparisons of the work involved more difficult.

To facilitate such comparisons, the operation count for each of these three methods is given using each relevant option. Total costs are therefore stated using either orthogonalization (ORTH) or elimination (ELIM) methods for computing Z, explicitly (EXP) or implicitly (IMP) defined Z, and matrix multiplication (MM) or finite differences (FD) in forming BZ. (The indicated abbreviations will be used to specify which options are used in a given algorithm.) Each method which has more than one option is accompanied by a graph showing the coefficient of n^3 as a function of $\beta = m/n$ for each option. Omitted from the tallies are steps whose operation counts are of degree lower than three (e.g., matrix-vector multiplication, solving triangular linear systems by back-substitution).

Method 1

Step	Cost
1. Factor $\begin{pmatrix} B & A \\ A^T & 0 \end{pmatrix}$	$(n+m)^3/6$
Total cost	$(1 + 3\beta + 3\beta^2 + \beta^3)n^3/6$

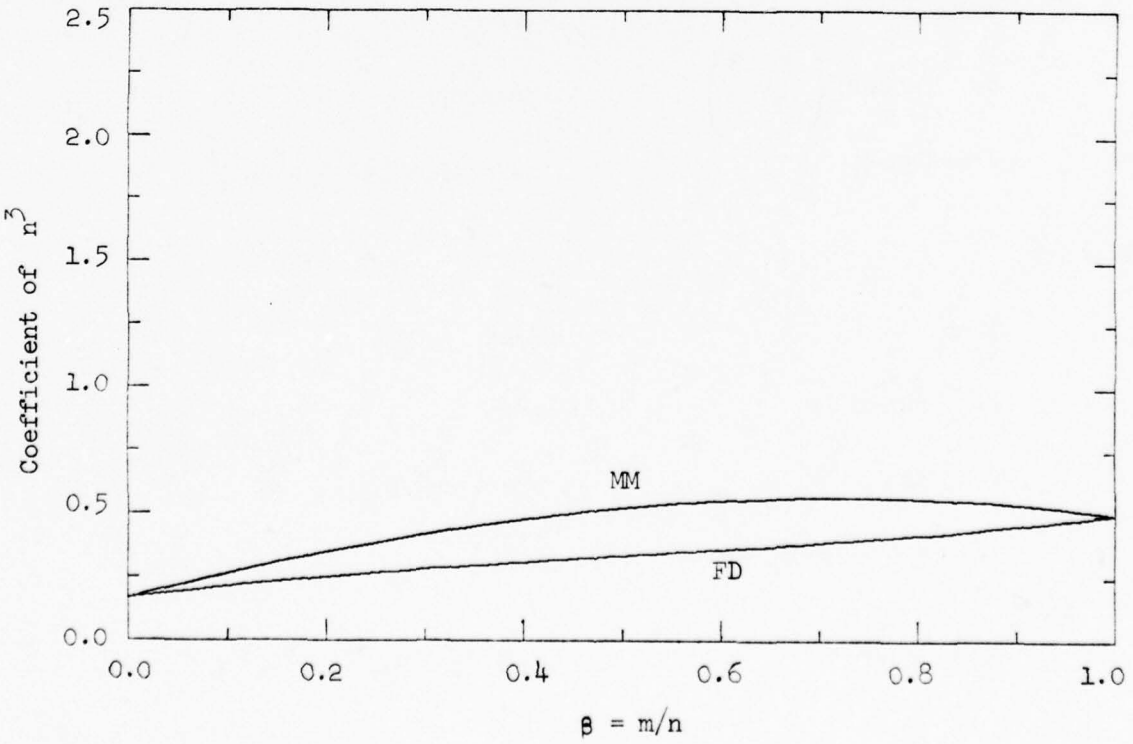
Method 2

Step	Cost
1. Factor B	$n^3/6$
2. Form $A^TB^{-1}A$	$n^2m + nm^2/2$
3. Factor $A^TB^{-1}A$	$m^3/6$
Total cost	$(1 + 6\beta + 3\beta^2 + \beta^3)n^3/6$

Method 3

Step	Cost
1. Factor A (ELIM)	$nm^2 - m^3/2$
2. Form Z^TBZ (MM)	$3n^2m/2 - 2nm^2 + m^3/2$
(FD)	$n^2m - 2nm^2 + m^3$
3. Factor Z^TBZ	$(n-m)^3/6$
Total cost (MM)	$(1 + 6\beta - 3\beta^2 - \beta^3)n^3/6$
(FD)	$(1 + 3\beta - 3\beta^2 + 2\beta^3)n^3/6$

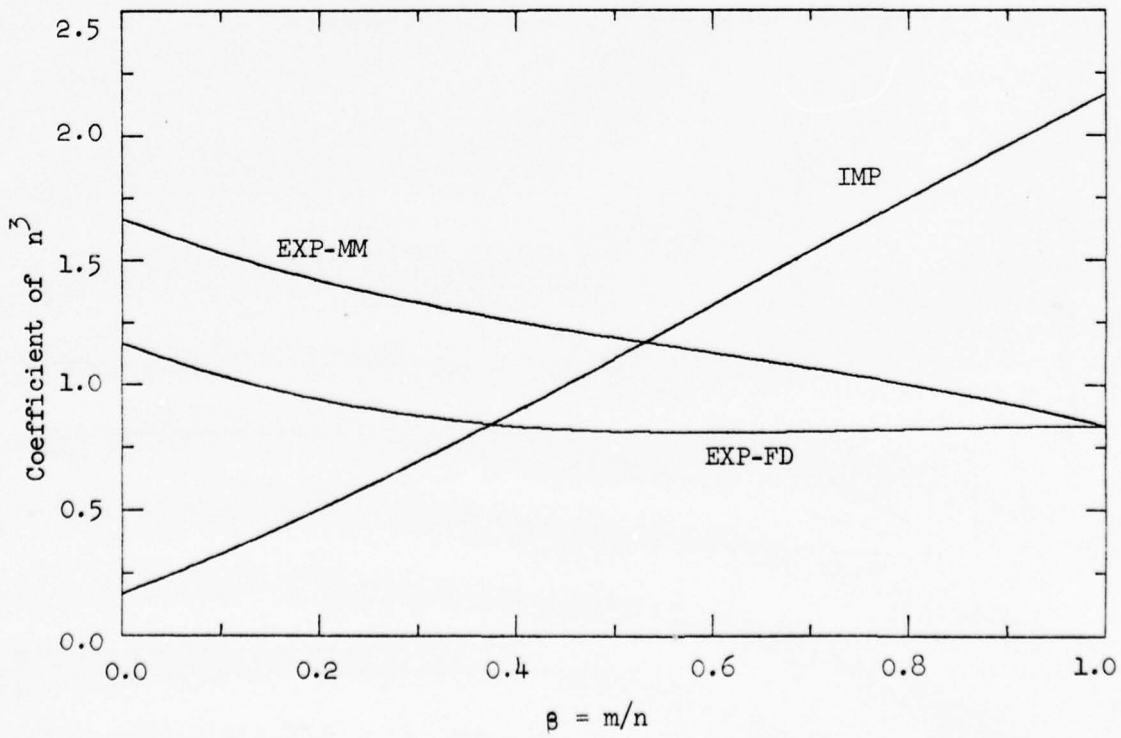
(Note: The linear systems (3.5) and (3.7) are solved using the factorization of A in Step 1.)



Method 4

Step		Cost
1. Factor A (ORTH)		$nm^2 - m^3/3$
2. Form Z (EXP)		$n^2m - nm^2/2 - m^3/3$
3. Form Z^TBZ	(EXP-MM)	$3n^3/2 - 2n^2m + nm^2/2$
	(EXP-FD)	$n^3 - 2n^2m + nm^2$
	(IMP)	$2n^2m - nm^2/2$
4. Factor Z^TBZ		$(n-m)^3/6$
Total cost	(EXP-MM)	$(10 - 9\beta + 9\beta^2 - 5\beta^3)n^3/6$
	(EXP-FD)	$(7 - 9\beta + 12\beta^2 - 5\beta^3)n^3/6$
	(IMP)	$(1 + 9\beta + 6\beta^2 - 3\beta^3)n^3/6$

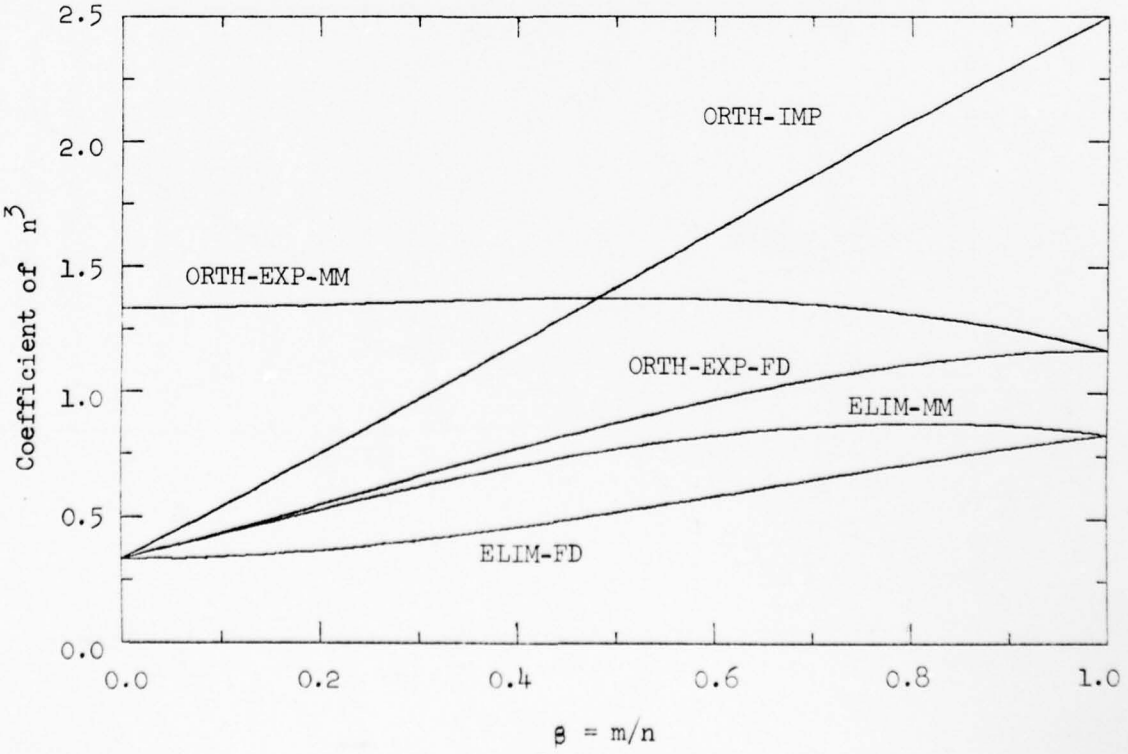
(Note: The linear systems (3.8) and (3.10) are solved using the factorization of A in Step 1.)



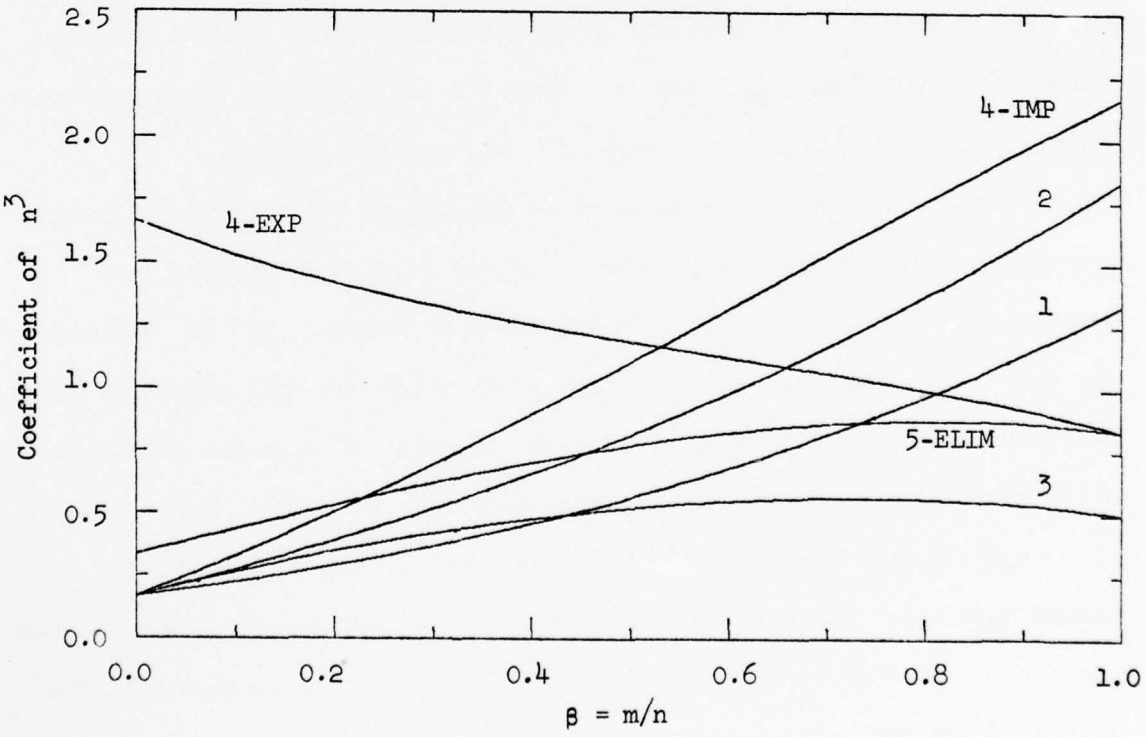
Method 5

Step		Cost
1. Factor A	(ELIM)	$nm^2 - m^3/2$
	(ORTH)	$nm^2 - m^3/3$
2. Form Z	(ORTH)	$n^2m - nm^2/2 - m^3/3$
3. Form Z^TB	(ELIM-MM)	$n^2m - nm^2$
	(ORTH-EXP-MM)	$n^3 - n^2m$
	(ORTH-IMP)	$2n^2m - nm^2/2$
4. Factor $\begin{pmatrix} Z^TB \\ A^T \end{pmatrix}$		$n^3/3$
Total cost	(ELIM-MM)	$(2 + 6\beta - 3\beta^3)n^3/6$
	(ELIM-FD)	$(2 + 6\beta^2 - 3\beta^3)n^3/6$
	(ORTH-EXP-MM)	$(8 + 3\beta^2 - 4\beta^3)n^3/6$
	(ORTH-EXP-FD)	$(2 + 6\beta + 3\beta^2 - 4\beta^3)n^3/6$
	(ORTH-IMP)	$(2 + 12\beta + 3\beta^2 - 2\beta^3)n^3/6$

(Note: The linear system (3.7) is solved using the factorization of A in Step 1.)



Since the finite difference option for forming BZ is unavailable in Methods 1 and 2, the most nearly comparable case among all five methods is the use of matrix multiplication by an explicitly available B. For direct comparison, this case is illustrated for all five methods in the following graph. Method 4 is shown with both EXP and IMP options. Method 5 is shown with the ELIM option, since this is the most efficient.



The following conclusions may be drawn. Method 1 is surprisingly competitive considering that it takes no advantage of the structure of system (3.2) other than symmetry. Method 2 is less efficient than Method 1 across the full range of β and is also less stable numerically. Method 3 is clearly the method of choice for all but the smallest values of β if efficiency is the sole criterion. Method 4 is reasonably competitive in efficiency only if a transition is made from implicit to explicit Z at about $\beta \approx \frac{1}{2}$. Method 5 is surprisingly competitive considering that it does not even take advantage of symmetry. In particular, Method 5 is more efficient than Methods 1, 2, and 4 for large β. However, Method 5 is clearly inferior to Method 3 and probably no more stable. These conclusions are valid only for relatively large n, when the highest-order terms are truly dominant. For small n, efficiency is not a critical issue.

Overall efficiency depends on other factors in addition to algebraic overhead. For example, the potential savings in function evaluations resulting from the use of finite differences in forming BZ may outweigh any gain which might be had with a method in which the full matrix B must be formed by finite differences. Indeed, the cost of function evaluations often dwarfs in importance questions of algebraic overhead. Moreover, efficiency is far from being the only issue which must be considered: a method must be accurate, stable and robust to be useful in a general purpose algorithm. In the context of Algorithm 3.1 it is also important that the method chosen for solving system (3.2) make available the range-space and null-space components of the Newton step s for use in the separate line searches, and also the matrix Z^TBZ is needed explicitly in the convergence test. When all these factors are taken into account, Method 4 is seen to

be a stable and reasonably efficient approach to solving system (3.2) which fits in well with the other aspects of Algorithm 3.1.

BIBLIOGRAPHY

Aasen, J. O. (1971). "On the Reduction of a Symmetric Matrix to Tridiagonal Form," BIT, vol. 11, pp. 233-242.

Abadie, J. and Carpentier, J. (1969). "Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints," Optimization (R. Fletcher, ed.), pp. 37-47, Academic Press, New York.

Akaike, H. (1959). "On a Successive Transformation of Probability Distribution and Its Application to the Analysis of the Optimum Gradient Method," Ann. Inst. Stat. Math. Tokyo, vol. 11, pp. 1-16.

Armijo, L. (1966). "Minimization of Functions Having Lipschitz-Continuous First Partial Derivatives," Pacific J. Math., vol. 16, pp. 1-3.

Asaadi, J. (1973). "A Computational Comparison of Some Non-Linear Programs," Math. Prog., vol. 4, pp. 144-154.

Avriel, M. (1976). Nonlinear Programming: Analysis and Methods, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Bard, Y. and Greenstadt, J. L. (1969). "A Modified Newton Method for Optimization with Equality Constraints," Optimization (R. Fletcher, ed.), pp. 299-306, Academic Press, New York.

Barwell, V. and George, A. (1976). "A Comparison of Algorithms for Solving Symmetric Indefinite Systems of Linear Equations," ACM Trans. Math. Software, vol. 2, pp. 242-251.

Bellman, R. (1970). Introduction to Matrix Analysis, Second Edition, McGraw-Hill Book Co., New York.

Ben-Israel, A. (1965). "A Modified Newton-Raphson Method for the Solution of Systems of Equations," Israel J. Math., vol. 3, pp. 94-98.

Bertsekas, D. P. (1976). "Multiplier Methods: A Survey," Automatica, vol. 12, pp. 133-145.

Biggs, M. C. (1972). "Constrained Minimization Using Recursive Equality Quadratic Programming," Numerical Methods for Non-Linear Optimization (F. A. Lootsma, ed.), pp. 411-428, Academic Press, New York.

Biggs, M. C. (1975). "Constrained Minimization Using Recursive Quadratic Programming: Some Alternative Subproblem Formulations," Towards Global Optimization (L. C. W. Dixon and G. P. Szegö, eds.), pp. 341-349, North-Holland Publishing Co., Amsterdam.

Blue, J. L. (1976). Solving Systems of Nonlinear Equations, Bell Laboratories Computing Science Technical Report No. 50.

Box, M. J. (1966). "A Comparison of Several Current Optimization Methods, and the Use of Transformations in Constrained Problems," Comput. J., vol. 9, pp. 67-77.

Bracken, J. and McCormick, G. P. (1968). Selected Applications of Non-linear Programming, John Wiley and Sons, New York.

Brent, R. P. (1973). Algorithms for Minimization without Derivatives, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Broyden, C. G. (1965). "A Class of Methods for Solving Nonlinear Simultaneous Equations," Math. Comput., vol. 19, pp. 577-593.

Bunch, J. R. and Kaufman, L. (1977a). "Some Stable Methods for Calculating Inertia and Solving Symmetric Linear Systems," Math. Comput., vol. 31, pp. 163-179.

Bunch, J. R. and Kaufman, L. (1977b). Indefinite Quadratic Programming, Bell Laboratories Computing Science Technical Report No. 61.

Bunch, J. R., Kaufman, L. and Parlett, B. N. (1976). "Decomposition of a Symmetric Matrix," Numer. Math., vol. 27, pp. 95-109.

Bunch, J. R. and Parlett, B. N. (1971). "Direct Methods for Solving Symmetric Indefinite Systems of Linear Equations," SIAM J. Numer. Anal., vol. 8, pp. 639-655.

Bus, J. C. P. (1975). A Comparative Study of Programs for Solving Nonlinear Equations, Mathematisch Centrum Report NW 25/75, Amsterdam.

Businger, P. A. and Golub, G. H. (1965). "Linear Least Squares Solutions by Householder Transformations," Numer. Math., vol. 7, pp. 269-276.

Buys, J. D. (1972). Dual Algorithms for Constrained Optimization Problems, Ph.D. Dissertation, University of Leiden.

Carroll, C. W. (1961). "The Created Response Surface Technique for Optimizing Nonlinear Restrained Systems," Operations Research, vol. 9, pp. 169-184.

Colville, A. R. (1968). A Comparative Study on Nonlinear Programming Codes, IBM New York Scientific Center Report No. 320-2949.

Courant, R. (1943). "Variational Methods for the Solution of Problems of Equilibrium and Vibrations," Bull. Amer. Math. Soc., vol. 49, pp. 1-23.

Curry, H. B. (1944). "The Method of Steepest Descent for Non-Linear Minimization Problems," Quart. Appl. Math., vol. 2, pp. 258-261.

Davidon, W. C. (1959). Variable Metric Method for Minimization, Argonne National Laboratory Report ANL-5990.

Dennis, J. E. (1971). "Algorithms for Nonlinear Problems which Use Discrete Approximations to Derivatives," Proc. ACM Natl. Conf., vol. 26, pp. 446-456.

Dennis, J. E., Gay, D. M., and Welsch, R. E. (1977). An Adaptive Nonlinear Least-Squares Algorithm, National Bureau of Economic Research Working Paper No. 196.

Dennis, J. E. and Mei, H. H.-W. (1975). An Unconstrained Optimization Algorithm which Uses Function and Gradient Values, Cornell University Dept. of Computer Science Report TR 75-246.

Dennis, J. E. and Moré, J. J. (1977). "Quasi-Newton Methods, Motivation and Theory," SIAM Rev., vol. 19, pp. 46-89.

Dixon, L. C. W. (1975). "Nonlinear Programming: A View of the State of the Art," Towards Global Optimization (L. C. W. Dixon and G. P. Szegö, eds.), pp. 320-340, North-Holland Publishing Co., Amsterdam.

Fiacco, A. V. and McCormick, G. P. (1968). Nonlinear Programming: Sequential Unconstrained Minimization Techniques, John Wiley and Sons, Inc., New York.

Fletcher, R. (1968). "Generalized Inverse Methods for the Best Least Squares Solution of Systems of Nonlinear Equations," Comput. J., vol. 10, pp. 392-399.

Fletcher, R. (1971). A Modified Marquardt Subroutine for Non-Linear Least Squares, Atomic Energy Research Establishment Report AERE-R 6799, Harwell, England.

Fletcher, R. (1975). "An Ideal Penalty Function for Constrained Optimization," J. Inst. Math. Appl., vol. 15, pp. 319-342.

Fletcher, R. (1977). "Methods for Solving Nonlinearly Constrained Optimization Problems," The State of the Art in Numerical Analysis (D. A. H. Jacobs, ed.), pp. 365-407, Academic Press, New York.

Fletcher, R. and Powell, M. J. D. (1963). "A Rapidly Convergent Descent Method for Minimization," Comput. J., vol. 6, pp. 163-168.

Fletcher, R. and Freeman, T. L. (1975). A Modified Newton Method for Minimization, University of Dundee Dept. of Mathematics Report No. 7.

Forsythe, G. E. and Moler, C. B. (1967). Computer Solution of Linear Algebraic Systems, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Gantmacher, F. R. (1959). The Theory of Matrices, Chelsea Publishing Co., New York.

Gill, P. E., Golub, G. H., Murray, W. and Saunders, M. A. (1974). "Methods for Modifying Matrix Factorizations," Math. Comput., vol. 28, pp. 505-535.

Gill, P. E. and Murray, W. (1972). Two Methods for the Solution of Linearly Constrained and Unconstrained Optimization Problems, National Physical Laboratory Report NAC 25.

Gill, P. E. and Murray, W. (1974a). Numerical Methods for Constrained Optimization, Academic Press, New York.

Gill, P. E. and Murray, W. (1974b). Safeguarded Steplength Algorithms for Optimization Using Descent Methods, National Physical Laboratory Report NAC 37.

Gill, P. E. and Murray, W. (1976). Algorithms for the Solution of the Non-Linear Least-Squares Problem, National Physical Laboratory Report NAC 71.

Gill, P. E. and Murray, W. (1977). The Computation of Lagrange-Multiplier Estimates for Constrained Minimization, National Physical Laboratory Report NAC 77.

Glad, T. (1976). Constrained Optimization Using Multiplier Methods with Applications to Control Problems, Lund Inst. Tech. Dept. of Automatic Control Report 7603.

Gleyzal, A. N. (1959). "Solution of Nonlinear Equations," Quart. Appl. Math., vol. 17 (1959), pp. 95-96.

Goldfeld, S. M., Quandt, R. E. and Trotter, H. F. (1966). "Maximization by Quadratic Hill-Climbing," Econometrica, vol. 34, pp. 541-551.

Goldstein, A. A. (1962). "Cauchy's Method of Minimization," Numer. Math., vol. 4, pp. 146-150.

Goldstein, A. A. and Price, J. F. (1967). "An Effective Algorithm for Minimization," Numer. Math., vol. 10, pp. 184-189.

Golub, G. H. (1965). "Numerical Methods for Solving Linear Least Squares Problems," Numer. Math., vol. 7, pp. 206-216.

Golub, G. H., Klema, V. and Stewart, G. W. (1976). Rank Degeneracy and Least Squares Problems, Stanford University Computer Science Dept. Report STAN-CS-76-559.

Golub, G. H. and Pereyra, V. (1976). "Differentiation of Pseudoinverses, Separable Nonlinear Least Square Problems and Other Tales," Generalized Inverses and Applications (M. Z. Nashed, ed.), pp. 303-324, Academic Press, New York.

Golub, G. H. and Reinsch, C. (1970). "Singular Value Decomposition and Least Squares Solutions," Numer. Math., vol. 14, pp. 403-420.

Golub, G. H. and Saunders, M. A. (1969). Linear Least Squares and Quadratic Programming, Stanford University Computer Science Dept. Report CS 134. Also appeared in Integer and Nonlinear Programming (J. Abadie, ed.), pp. 229-256, North-Holland Publishing Co., Amsterdam, 1970.

Greenstadt, J. (1967). "On the Relative Efficiencies of Gradient Methods," Math. Comput. vol. 21, pp. 360-367.

Haarhoff, P. C. and Buys, J. D. (1970). "A New Method for the Optimization of a Nonlinear Function Subject to Nonlinear Constraints," Comput. J., vol. 13, pp. 178-184.

Han, S. P. (1976). "Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems," Math. Prog., vol. 11, pp. 263-282. See also Cornell University Dept. of Computer Science Reports TR 75-233, 250, 252 and 257.

Hardaway, R. H. (1968). "An Algorithm for Finding a Solution of Simultaneous Nonlinear Equations," AFIPS Conf. Proc., vol. 33, Part 1, pp. 105-113.

Hebden, M. D. (1973). An Algorithm for Minimization Using Exact Second Derivatives, Atomic Energy Research Establishment Report AERE-TP.515, Harwell, England.

Hestenes, M. R. (1969). "Multiplier and Gradient Methods," J. Optimization Theory Appl., vol. 4, pp. 303-320.

Himmelblau, D. M. (1972). Applied Nonlinear Programming, McGraw-Hill Book Co., New York.

Householder, A. S. (1958). "Unitary Triangularization of a Nonsymmetric Matrix," J. Assoc. Comput. Mach., vol. 5, pp. 339-342.

Householder, A. S. (1964). The Theory of Matrices in Numerical Analysis, Blaisdell Publishing Co., New York.

Isaacson, E. and Keller, H. B. (1966). Analysis of Numerical Methods, John Wiley and Sons, Inc., New York.

Jones, A. (1970). "Spiral--A New Algorithm for Non-Linear Parameter Estimation Using Least Squares," Comput. J., vol. 13, pp. 301-308.

Karasalo, I. (1974). "A Criterion for Truncation of the QR-Decomposition Algorithm for the Singular Linear Least Squares Problem," BIT, vol. 14, pp. 156-166.

Kaufman, L. (1977). Application of Householder Transformations to a Sparse Matrix, Bell Laboratories Computing Science Technical Report.

Kelley, H. J., Denham, W. F., Johnson, I. L. and Wheatley, P. O. (1966). "An Accelerated Gradient Method for Parameter Optimization with Non-Linear Constraints," J. Astronaut. Sci., vol. 13, pp. 166-169.

Klein, B. (1955). "Direct Use of Extremal Principles in Solving Certain Optimizing Problems Involving Inequalities," Operations Research, vol. 3, pp. 168-175.

Lawson, C. L. and Hanson, R. J. (1974). Solving Least Squares Problems, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Levenberg, K. (1944). "A Method for the Solution of Certain Non-Linear Problems in Least Squares," Quart. Appl. Math., vol. 2, pp. 164-168.

Lootsma, F. A. (1972). "Penalty Function Performance of Several Unconstrained Minimization Techniques," Philips Res. Reports, vol. 27, pp. 358-385.

Leunberger, D. G. (1973). Introduction to Linear and Nonlinear Programming, Addison-Wesley Publishing Co., Reading, Massachusetts.

Mangasarian, O. L. (1976). "Unconstrained Methods in Nonlinear Programming," Nonlinear Programming (R. W. Cottle and C. E. Lemke, eds.), SIAM-AMS Proc., vol. 9, pp. 169-184, Amer. Math. Soc.

Marquardt, D. W. (1963). "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," J. Soc. Indust. Appl. Math., vol. 11, pp. 431-441.

McCormick, G. P. (1970). "A Second Order Method for the Linearly Constrained Nonlinear Programming Problem," Nonlinear Programming (J. B. Rosen, O. L. Mangasarian and K. Ritter, eds.), pp. 207-243, Academic Press, New York.

McCormick, G. P. (1976). Second Order Convergence Using a Modified Armijo Step-Size Rule for Function Minimization, George Washington University Institute for Management Science and Engineering Serial T-328.

Miele, A., Huang, H. Y. and Heideman, J. C. (1969). "Sequential Gradient-Restoration Algorithm for the Minimization of Constrained Functions--Ordinary and Conjugate Gradient Versions," J. Optimization Theory Appl., vol. 4, pp. 213-243.

Miele, A., Cragg, E. E., Iyer, R. R. and Levy, A. V. (1971). "Use of the Augmented Penalty Function in Mathematical Programming Problems," J. Optimization Theory Appl., vol. 8, pp. 115-153.

Miele, A., Moseley, P. E., Levy, A. V. and Coggins, G. M. (1972). "On the Method of Multipliers for Mathematical Programming Problems," J. Optimization Theory Appl., vol. 10, pp. 1-33.

Moré, J. J. (1977). "The Levenberg-Marquardt Algorithm: Implementation and Theory," Proc. 1977 Dundee Conf. Numer. Anal.

Murray, W. (1969). Constrained Optimization, National Physical Laboratory Report. MA 79.

Murray, W. (1972). "Failure, the Causes and Cures," Numerical Methods for Unconstrained Optimization (W. Murray, ed.), pp. 107-122, Academic Press, New York.

Nash, J. C. (1976). An Annotated Bibliography on Methods for Nonlinear Least Squares Computations Including Test Problems, Mary Nash Information Services, Vanier, Ontario, Canada.

O'Doherty, R. J. and Pierson, B. L. (1974). "A Numerical Study of Multiplier Methods for Constrained Parameter Optimization," Internat. J. Systems Sci., vol. 5, pp. 187-200.

Orden, A. (1964). "Stationary Points of Quadratic Functions under Linear Constraints," Comput. J., vol. 7, pp. 238-242.

Ortega, J. M. and Rheinboldt, W. C. (1970). Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York.

Parlett, B. N. and Reid, J. K. (1970). "On the Solution of a System of Linear Equations whose Matrix is Symmetric but not Definite," BIT, vol. 10, pp. 386-397.

Penrose, R. (1955), "A Generalized Inverse for Matrices," Proc. Combridge Philos. Soc., vol. 51, pp. 406-413.

Peters, G. and Wilkinson, J. H. (1970). "The Least Squares Problem and Pseudo-Inverses," Computer J., vol. 13, pp. 309-316.

Pierre, D. A. and Lowe, M. J. (1975). Mathematical Programming via Augmented Lagrangians, Addison-Wesley Publishing Co., Reading, Massachusetts.

Powell, M. J. D. (1969). "A Method for Nonlinear Constraints in Minimization Problems," Optimization (R. Fletcher, ed.), pp. 283-297, Academic Press, New York.

Powell, M. J. D. (1970a). "A Hybrid Method for Nonlinear Equations," Numerical Methods for Nonlinear Algebraic Equations (P. Rabinowitz, ed.), pp. 87-114, Gordon and Breach Science Publishers, New York.

Powell, M. J. D. (1970b). "A New Algorithm for Unconstrained Optimization," Nonlinear Programming (J. B. Rosen, O. L. Mangasarian and K. Ritter, eds.) pp. 31-65, Academic Press, New York.

Powell, M. J. D. (1976). "Algorithms for Nonlinear Constraints that Use Lagrangian Functions," Ninth Internat. Symp. Math. Prog., Budapest. See also University of Cambridge Reports DAMTP 77/NA 2 and 3.

Robinson, S. M. (1972). "A Quadratically-Convergent Algorithm for General Nonlinear Programming Problems," Math. Prog., vol. 3, pp. 145-156.

Robinson, S. M. (1974). "Perturbed Kuhn-Tucker Points and Rates of Convergence for a Class of Nonlinear-Programming Algorithms," Math. Prog., vol. 7, pp. 1-16.

Rockafellar, R. T. (1973). "A Dual Approach to Solving Nonlinear Programming Problems by Unconstrained Optimization," Math. Prog., vol. 5, pp. 354-373.

Rosen, J. B. (1960). "The Gradient Projection Method for Nonlinear Programming, Part I. Linear Constraints," J. Soc. Indust. Appl. Math., vol. 8, pp. 181-217.

Rosen, J. B. (1961). "The Gradient Projection Method for Nonlinear Programming, Part II. Nonlinear Constraints," J. Soc. Indust. Appl. Math., vol. 9, pp. 514-532.

Rosen, J. B. (1964). "Minimum and Basic Solutions to Singular Linear Systems," J. Soc. Indust. Appl. Math., vol. 12, pp. 156-162.

Rosen, J. B. (1977). Two-Phase Algorithm for Nonlinear Constraint Problems, University of Minnesota Computer Science Dept. Tech. Report 77-8.

Rosen, J. B. and Kreuser, J. (1972). "A Gradient Projection Algorithm for Non-Linear Constraints," Numerical Methods for Non-Linear Optimization (F. A. Lootsma, ed.), pp. 297-300, Academic Press, New York.

Rosen, J. B. and Suzuki, S. (1965). "Construction of Nonlinear Programming Test Problems," Comm. ACM, vol. 8, p. 113.

Sargent, R. W. H. (1974). "Reduced-Gradient and Projection Methods for Nonlinear Programming," Numerical Methods for Constrained Optimization (P. E. Gill and W. Murray, eds.), pp. 149-174, Academic Press, New York.

Sorensen, D. C. (1977). Updating the Symmetric Indefinite Factorization with Applications in a Modified Newton's Method, Argonne National Laboratory Report ANL-77-49.

Stewart, G. W. (1967). "A Modification of Davidon's Minimization Method to Accept Difference Approximations of Derivatives," J. Assoc. Comput. Mach., vol. 14, pp. 72-83.

Stewart, G. W. (1973). Introduction to Matrix Computations, Academic Press, New York.

Tapia, R. A. (1974). "A Stable Approach to Newton's Method for General Mathematical Programming Problems in R^n," J. Optimization Theory Appl., vol. 14, pp. 453-476.

Tapia, R. A. (1977). "Diagonalized Multiplier Methods and Quasi-Newton Methods for Constrained Optimization," J. Optimization Theory Appl., vol. 22, pp. 135-194.

Van Loan, C. (1976). Lectures in Least Squares, Cornell University Dept. of Computer Science Report TR 76-279.

Welsch, R. E. and Becker, R. A. (1975). Robust Nonlinear Regression Using the Dogleg Algorithm, National Bureau of Economic Research Working Paper No. 76.

Wilkinson, J. H. (1965). The Algebraic Eigenvalue Problem, Oxford University Press, London.

Wilson, R. B. (1963). A Simplicial Algorithm for Concave Programming, Ph.D. Dissertation, Harvard University.

Wolfe, P. (1963). "Methods of Nonlinear Programming," Recent Advances in Mathematical Programming (R. L. Graves and P. Wolfe, eds.), pp. 67-86, McGraw-Hill Book Co., New York.

Wolfe, P. (1967). "Methods of Nonlinear Programming," Nonlinear Programming (J. Abadie, ed.), pp. 97-131, North-Holland Publishing Co., Amsterdam.

Wolfe, P. (1969). "Convergence Conditions for Ascent Methods," SIAM Rev., vol. 11, pp. 226-235.

Wolfe, P. (1971). "Convergence Conditions for Ascent Methods II: Some Corrections," SIAM Rev., vol. 13, pp. 185-188.

Wolfe, P. (1972). "On the Convergence of Gradient Methods under Constraint," IBM J. Res. Develop., vol. 16, pp. 407-411.

Wright, M. H. (1976). Numerical Methods for Nonlinearly Constrained Optimization, Stanford Linear Accelerator Center Report SLAC-193.

Zangwill, W. I. (1969). Nonlinear Programming: A Unified Approach, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Zoutendijk, G. (1960). Methods of Feasible Directions, Elsevier Publishing Co., Amsterdam.